UNIVERSIDADE DO VALE DO TAQUARI
CURSO DE ENGENHARIA DA COMPUTAÇÃO

SISTEMA DE RECOMENDAÇÃO DE EVENTOS UTILIZANDO O *FRAMEWORK* APACHE MAHOUT

Christian Rodolpho Bugs

Lajeado, novembro de 2019

Christian Rodolpho Bugs

SISTEMA DE RECOMENDAÇÃO DE EVENTOS UTILIZANDO O O *FRAMEWORK* APACHE MAHOUT

Monografia apresentada à disciplina de Trabalho de Conclusão de Curso II, do Curso de Engenharia da Computação, da Universidade do Vale do Taquari - UNIVATES, como parte da exigência para a obtenção do título de Bacharel em Engenharia da Computação

Orientador: Prof. Me. Willian Valmorbida

Lajeado, novembro de 2019

RESUMO

Atualmente, as instituições de ensino oferecem aos estudantes uma infinidade de eventos. Entretanto, devido à essa grande variedade, os alunos acabam tendo dificuldades para decidir no que se inscrever. Uma alternativa a este problema é a utilização de um Sistema de recomendação (SR), possibilitando a instituição de ensino ofertar aos alunos os melhores eventos de acordo com seu perfil e interesses. Posto isto, o objetivo principal deste trabalho foi desenvolver um sistema de recomendações de eventos, para a Univates. No processo de desenvolvimento do SR, foi utilizada a técnica de aprendizagem por máquina, sendo que os dados filtrados foram processados com o auxílio da *framework* Apache Mahout, através de filtragem colaborativa, que é a filtragem que busca perfis similares ao do usuário para criar recomendações. A fim de verificar a qualidade das recomendações geradas, foi realizada uma avaliação com usuários reais, sendo eles alunos e funcionários da instituição, a fim de gerar 3 recomendações. Com relação às avaliações feitas para as 3 recomendações geradas, o resultado de pontuação média de 7,78 demonstra a grande aceitação dos usuários ao sistema proposto. Os resultados demonstram que o objetivo geral de criar um sistema de recomendações de eventos utilizando a *framework* Apache Mahout foi atingido e que a sua implementação real é viável.

Palavras-chave: Sistema de recomendação. Eventos. Filtragem colaborativa. Aprendizagem por máquina. Apache Mahout.

ABSTRACT

Today, educational institutions offer students a multitude of events. However, because of this wide variety, students find it difficult to decide what to apply for. An alternative to this problem is the use of a Recommendation System (SR), enabling the educational institution to offer students the best events according to their profile and interests. That said, the main objective of this paper was to develop a system of event recommendations for Univates. In the SR development process, the machine learning technique was used, and the filtered data were processed with the aid of the Apache Mahout framework, through collaborative filtering, which is the filtering that seeks similar profiles to the user to create recommendations. . In order to verify the quality of the recommendations generated, an evaluation was conducted with real users, being students and employees of the institution ~, in order to generate 3 recommendations. Regarding the evaluations made for the 3 recommendations generated, the average score result of 7.78 shows the great acceptance of users to the proposed system. The results show that the overall objective of creating an event recommendation system using the Apache Mahout framework has been achieved and that its actual implementation is feasible.

Keywords: Recommendation system. Events. Collaborative filtering. Machine learning. Apache Mahout.

LISTA DE FIGURAS

Figura 1 – Arquitetura de um sistema de recomendação 14
Figura 2 – Filtragem baseada em conteúdo 16
Figura 3 – Exemplo de FBC utilizada pela empresa *Netshoes* 17
Figura 4 – Filtragem colaborativa 18
Figura 5 – Exemplo de FC utilizada por site IMDb 19
Figura 6 – Exemplo de FC utilizada pela rede social Instagram 19
Figura 7 – Exemplo de FH utilizada pelo site de receitas Tudo Gosto 21
Figura 8 – Fluxograma do projeto 33
Figura 9 – Modelo de dados no Apache Mahout 34
Figura 10 – Modelo de casos de uso 37
Figura 11 – Diagrama de estados 38
Figura 12 – Tabelas para controle de acesso do *Framework* 39
Figura 13 – Modelo de banco de dados do protótipo 41
Figura 14 – Coeficiente de Tanimoto 43
Figura 15 – Precisão e *recall* 44
Figura 16 – Teste de precisão e *recall* 45
Figura 17 – Sistema de recomendações 47
Figura 18 – Listagem de conexões 48
Figura 19 – Cadastro de conexões 48
Figura 20 – Cadastro de fontes de dados 49
Figura 21 – Função buscaRecomendacoes 50
Figura 22 – Chamada via SOAP 51
Figura 23 – Questionário de avaliação 52

LISTA DE TABELAS

Tabela 1 – Requisitos funcionais 35
Tabela 2 – Requisitos não-funcionais 36
Tabela 3 – Especificação da estrutura de controle de acesso do *Framework* Adianti 40
Tabela 4 – Teste de precisão e *recall* 46

LISTA DE GRÁFICOS

Gráfico 1 – Respostas da questão 2 54
Gráfico 2 – Resultados da recomendação 1 55
Gráfico 3 – Resultados da recomendação 2 55
Gráfico 4 – Resultados da recomendação 3 56

LISTA DE ABREVIATURAS

API:	*Application Programming Interface*
ASF:	*Apache Software Foundation*
ER:	Entidade Relacional
FBC:	Filtragem Baseada em conteúdo
FC:	Filtragem Colaborativa
FH:	Filtragem Híbrida
MCV:	*Model, View, Controller*
SR:	Sistema de Recomendação
SVM:	Máquinas de Vetor de Suporte
SOAP:	*Simple Object Access Protocol*
PHP:	*Hypertext Preprocessor Language*
HTML:	*Hypertext Markup Language*
CSS:	*Cascading Style Sheets*
JDBC:	*Java Database Connectivity*

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Objetivo Geral 11
1.1.1 Objetivos Específicos 12
1.2 Organização da Pesquisa 12
2 REFERENCIAL TEÓRICO 13
2.1 Sistemas de Recomendação 13
2.1.1 Técnicas de Filtragem 15
2.1.1.1 Filtragem Baseada em Conteúdo 15
2.1.1.2 Filtragem Colaborativa 17
2.1.1.3 Filtragem Híbrida 20
2.1.2 Ferramentas de Filtragem 21
2.1.2.1 Aprendizagem por Máquina 21
2.1.2.1.1 Aprendizagem Supervisionada 22
2.1.2.1.2 Aprendizagem Não-Supervisionada 23
2.2 Trabalhos Relacionados 24
3 METODOLOGIA 27
3.1 Tipo de Pesquisa 27
3.2 Ferramentas Utilizadas ..... Erro! Indicador não definido.
3.2.1 Apache 28
3.2.2 Apache Mahout 28
3.2.3 PHP 29
3.2.4 Adianti *Framework* 29
3.2.5 PostgreSQL 30
4 ESPECIFICAÇÕES DO PROJETO 31
4.1 Visão Geral do Projeto 31
4.2 Requisitos 34
4.2.1 Requisitos Funcionais 34
4.2.2 Requisitos Não-Funcionais 35
4.3 Modelo de Casos de Uso 36
4.4 Diagrama de Estados 37
5 IMPLEMENTAÇÃO 39
5.1 Estrutura do Banco de Dados 39
5.2 Implementação do Sistema de Recomendação 41
5.3 Implementação da Interface de Gestão de Fontes de Dados 47

5.4 Implementação da Interface de Avaliação 50
6 RESULTADOS E DISCUSSÃO 53
7 CONCLUSÕES 58
7.1 Trabalhos Futuros 59
REFERÊNCIAS 60

1 INTRODUÇÃO

Atualmente as instituições de ensino oferecem aos estudantes muitas opções de eventos, como palestras, shows, oficinas, cursos e congressos. Apesar disso ser benéfico, os alunos acabam enfrentando dificuldade para decidir no que se inscrever ou até de encontrar eventos que realmente interessam a ele.

A partir desta problemática, este trabalho apresenta a criação de um sistema para realizar a recomendação dos eventos, por meio de uma filtragem colaborativa, baseada no usuário. Com a disponibilização de recomendações automáticas, busca-se personalizar a experiência do usuário, direcionando para ele itens dos quais haja provável interesse, melhorando assim a divulgação desses eventos.

Os Sistemas de Recomendação (SR) vem sendo muito utilizados por empresas de todos os ramos; seu maior objetivo é ofertar ao consumidor itens previamente selecionados de acordo com seus gostos e necessidades, aumentando assim as chances de vendas e fechamento de negócios (AGGARWAL, 2016).

O processo de um sistema de recomendações começa na coleta de dados, que podem ser fornecidos pelo usuário de maneira explícita, ou coletados de maneira implícita através de seus históricos de compras, cliques, páginas visitadas, entre outros (RICCI; ROKACH; SHAPIRA, 2015). Esta etapa é fundamental, pois com quanto mais dados o SR é alimentado, mais ele se torna inteligente e as recomendações mais eficazes. Além de coletar os dados do usuário, ao implementar um SR, é preciso definir como os itens a serem recomendados serão filtrados, ou seja, como estes itens serão buscados na base de dados e relacionados com o perfil do usuário (ALVAREZ et al., 2016).

É na filtragem de dados que são selecionadas as recomendações relevantes para os usuários. Dentre os filtros mais comumente utilizados estão o baseado em conteúdo, o colaborativo e o híbrido (AGUIAR; FECHINE; COSTA, 2018). O filtro baseado em conteúdo recomenda ao usuário itens semelhantes aos que ele já adquiriu ou demonstrou interesse, o colaborativo faz recomendações baseadas no conhecimento das relações dos usuários com os itens e o híbrido é uma mistura de ambos (MELVILLE; SHINDHWANI, 2017).

Para o desenvolvimento deste projeto, foi utilizada a filtragem colaborativa, tendo em vista que essa estrutura de recomendação não analisa o tipo do item e nem seus atributos, não utilizando nenhuma característica específica dele para gerar a recomendação. A aplicação da filtragem colaborativa foi adequada, já que permite que os eventos sejam recomendados de acordo com os gostos e preferências de alunos com usuários similares. Como o protótipo do sistema de recomendação requer uma programação sofisticada, foi utilizada técnica de aprendizagem por máquina e os dados filtrados foram processados com o auxílio da biblioteca Apache Mahout.

A melhor divulgação dos serviços oferecidos pela instituição, pode vir a se tornar um elemento potencializador para o número de inscritos, uma vez que não será mais necessário buscar pelos eventos que estão acontecendo na sua área já que estes irão ao encontro dos consumidores, aumentando a possibilidade de captação através de serviços que podem ser disponibilizados quanto para alunos quanto para a comunidade.

Desta forma, o objetivo principal deste trabalho é desenvolver um sistema de recomendações de eventos para a Universidade Univates, utilizando a biblioteca Apache Mahout, criando assim, uma estratégia que irá beneficiar tanto os alunos, que receberão recomendações personalizadas, tanto a instituição que poderá ter um aumento no número de participantes em seus eventos.

1.1 Objetivo Geral

O objetivo geral deste projeto é oferecer recomendações de eventos personalizadas aos estudantes da Univates através da criação de um sistema de recomendações utilizando a *framework* Apache Mahout.

1.1.1 Objetivos Específicos

Para este trabalho, foram estabelecidos os seguintes objetivos específicos:

- Realizar um levantamento bibliográfico acerca do tema proposto;
- Estudar o *framework* de recomendações Apache Mahout;
- Modelar um sistema de recomendação colaborativa utilizando o *framework* Apache Mahout;
- Desenvolver um protótipo baseado no modelo proposto;
- Experimentar e avaliar os resultados obtidos na utilização do sistema de recomendação.

1.2 Organização da Pesquisa

As seções a seguir deste trabalho estão organizadas como segue:

- O capítulo 1 apresenta a introdução acerca do tema proposto, os objetivos e estrutura do trabalho;
- No capítulo 2 é apresentado o referencial teórico que serve como embasamento para o desenvolvimento do trabalho, nele é feito um maior aprofundamento sobre os sistemas de recomendação e as técnicas utilizadas para desenvolve-los;
- No capítulo 3 é abordada a metodologia utilizada para a realização do projeto, incluindo as ferramentas que serão utilizadas para construção do SR, sendo elas: Apache, Apache Mahout, PHP, Adianti *Framework* e PostgreSQL;
- O capítulo 4 apresenta uma visão geral da solução proposta, especificações de requisitos e os artefatos gerados;
- No capítulo 5 estão expostos os resultados através de gráficos e uma discussão completa de todos os resultados obtidos no trabalho;
- O capítulo 6 apresenta as conclusões e trabalhos futuros.

.

2 REFERENCIAL TEÓRICO

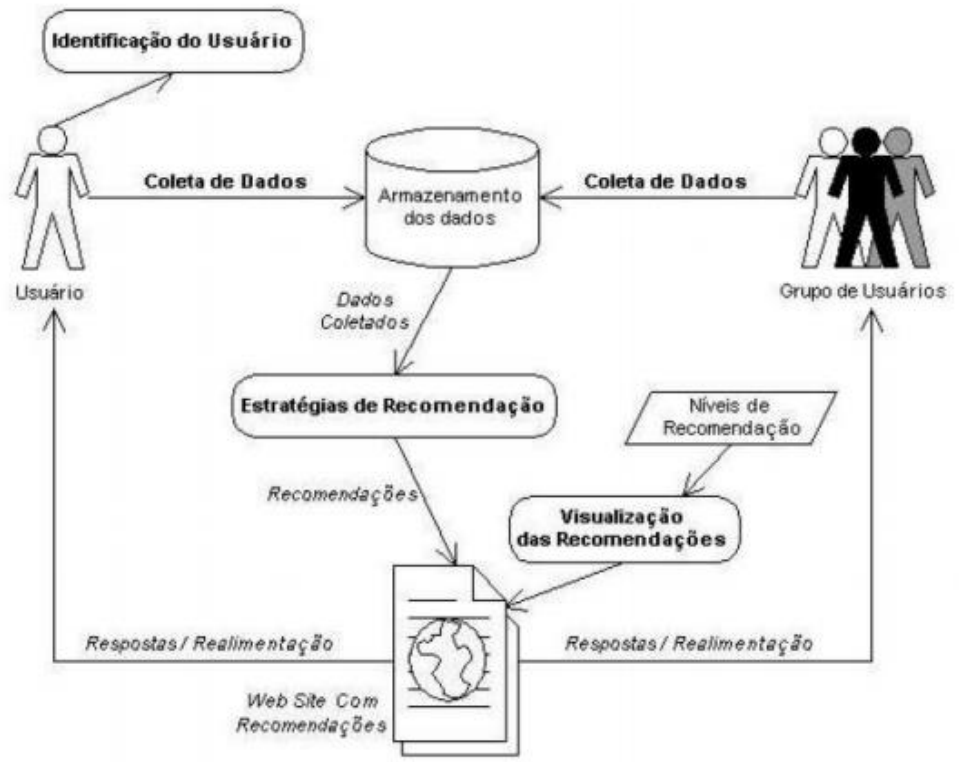
Na presente seção é apresentado um breve referencial teórico sobre os Sistemas de Recomendação, abordando conceitos, técnicas de implementação, vantagens e limitações.

2.1 Sistemas de Recomendação

Com o grande número de eventos, palestras, cursos e congressos oferecidos pelas instituições de ensino os estudantes acabam tendo dificuldade para decidir no que se inscrever e participar. Neste sentido, os Sistemas de Recomendação (SR) podem ser utilizados como uma ferramenta para recomendar conteúdos com base nas necessidades ou gosto do estudante (ROLIM, 2017). Atualmente os SR são amplamente utilizados na web, tornando possível recomendar ao usuário filmes, produtos, restaurantes, viagens e uma infinidade de itens com base nos sites em que navegaram (AGGARWAL, 2016).

Um dos intuitos de um SR é fazer com que o cliente sinta-se especial e bem atendido, devido ao cuidado que a empresa teve em lhe indicar produtos e serviços adequados às suas preferências pessoais (ALVAREZ et al., 2016). A Figura 1 ilustra a estrutura de um SR baseada no modelo de Schafer. Segundo Schafer, Konstan, e Riedl (2000) a estrutura de um SR é dividida em quatro processos: identificação do usuário, coleta de informações, estratégias de recomendação e visualização das recomendações.

Figura 1 – Arquitetura de um sistema de recomendação



Fonte: Schafer, Konstan, e Riedl (2000).

A identificação do usuário não é um processo obrigatório em um SR, mas quando é utilizado, deve ser o primeiro passo a ser feito. Com o usuário identificado, é possível customizar o sistema de acordo com as suas característica, podendo gerar recomendações personalizadas (SCHAFER, 2000).

A coleta dos dados do usuário pode ser feita de forma explícita, implícita ou inferência. Na forma explícita, os dados informados diretamente pelo usuário, como por exemplo, em um formulário onde ele informa suas preferências. Na forma implícita, as informações podem ser coletadas através de um monitoramento de navegação do usuário pelo site, por um histórico de compras, conteúdos vistos pelos usuários, etc. A coleta de dados por inferência consiste em descobrir o perfil de um usuário através da comparação entre padrões de comportamento de usuários similares. Após esta coleta de dados, são criadas as recomendações, com base na estratégia de recomendação definida pelo desenvolvedor do SR. A visualização das recomendações devem ser de fácil visualização e compreensão (RICCI; ROKACH; SHAPIRA, 2015).

Os Sistemas de recomendações são uma importante ferramenta para empresas de diferentes ramos pois identificam automaticamente conteúdos relevantes aos seus usuários conforme seus interesses e características. Na prática um SR faz uma relação eficaz entre o perfil do cliente em potencial com os serviços e produtos da empresa, para que possam ser oferecidos a ele somente itens que irão despertar seu interesse (AGGARWAL, 2016; CHENG et al., 2016).

As principais técnicas que são utilizadas para que o SR consiga identificar o tipo de perfil do usuário são a extração implícita e a extração explícita de informações (BARROS, 2014). A extração Implícita é feita através da coleta de dados de navegação do usuário no sistema, são extraídas informações como: menus de navegação, histórico de compras, itens adicionados a listas de desejos, entre outras. Já a extração explícita é feita através de informações informadas pelo próprio usuário, através de questionários, avaliações de produtos, entre outras formas (GUNAWARDANA; SHANI, 2015; RICCI; ROKACH; SHAPIRA, 2015).

2.1.1 Técnicas de Filtragem

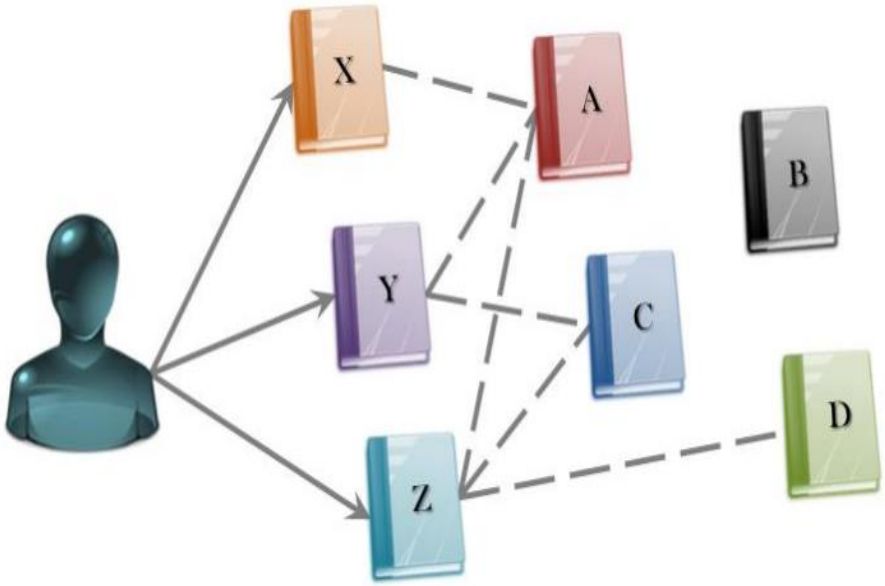
Com o usuário identificado e seus dados armazenados se faz necessário um bom sistema de filtragem de informações que permita definir os itens a serem recomendados (produtos, serviços, viagens, etc.). Basicamente existem três tipos de filtragem de informações que podem ser aplicadas em um SR, a baseada em conteúdo, a colaborativa e híbrida (GUY, 2015; MELVILLE; SHINDHWANI, 2017).

2.1.1.1 Filtragem Baseada em Conteúdo

A técnica de filtragem baseada em conteúdo (FBC) pode analisar os produtos e serviços já pesquisados ou adquiridos pelo usuário e então fazer a recomendação de conteúdo similar (AGUIAR; FECHINE; COSTA, 2018). A Figura 2, ilustra a técnica de Filtragem Baseada em Conteúdo, na ilustração o usuário leu os livros X, Y e Z e o livro B não possui nenhuma semelhança com os demais livros, sendo assim o livro B não será recomendado para este usuário; porém o livro A tem características semelhantes aos três

livros lidos, e acabará sendo recomendado; os livros C e D também possuem semelhança com ao menos um dos livros lidos, sendo assim, dependendo do nível de similaridade utilizado no sistema, eles também podem ser recomendados.

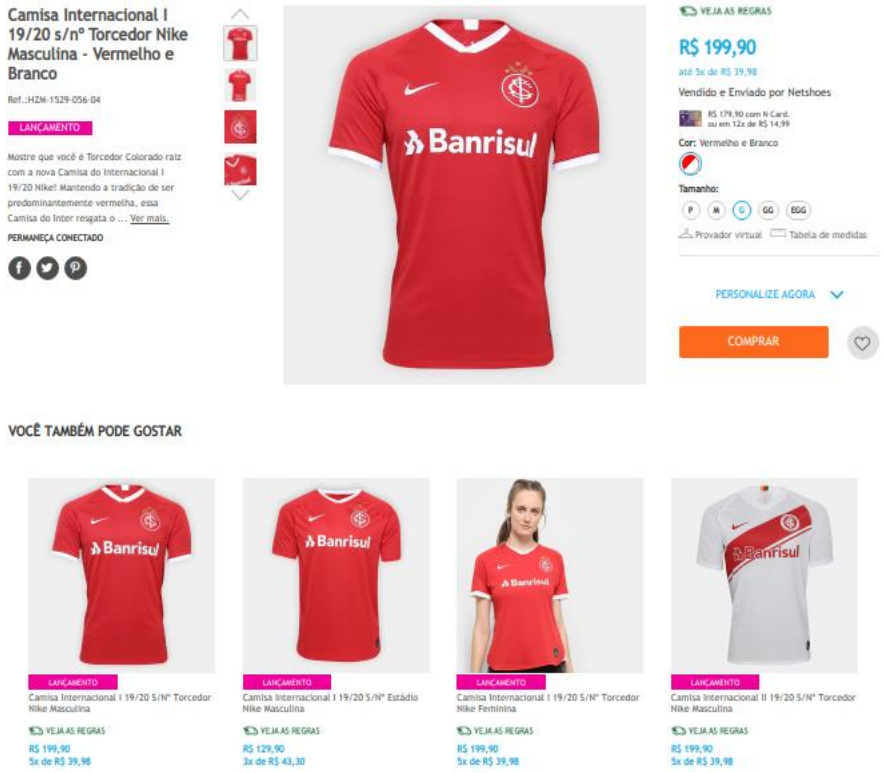
Figura 2 – Filtragem baseada em conteúdo



Fonte: Costa, Aguiar e Magalhães (2013).

Quando o usuário fornece informações como respostas a um questionário, preenche dados de perfil, pesquisa por itens, adiciona itens na sacola de compras. Os sistemas de FBC filtram esses dados para verificar o que possivelmente o interessaria (AGUIAR; FECHINE; COSTA, 2018). Na Figura 3, é possível ver um exemplo de FCB, o site de vendas de itens esportivos *Netshoes* recomenda ao usuário 4 itens similares ao que ele demonstrou interesse em comprar.

Figura 3 – Exemplo de FBC utilizada pela empresa *Netshoes*



Fonte: *Netshoes* (2019, texto digital).

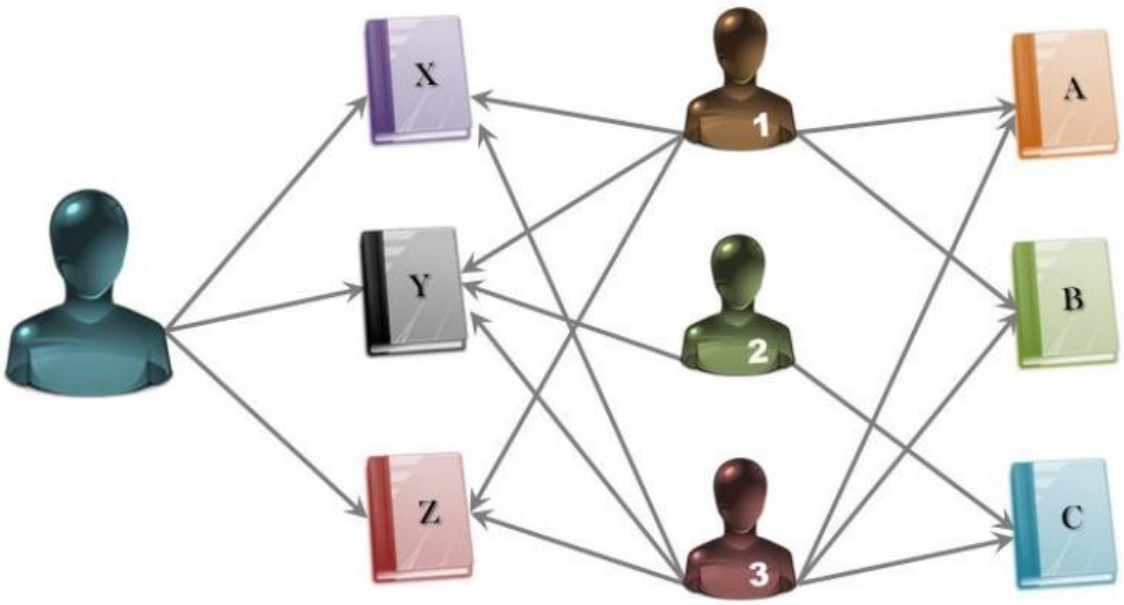
Dentre as vantagens da FBC pode-se destacar a possibilidade de encontrar bons resultados para usuários incomuns, além das recomendações que independem do número de usuários e que melhoram com o tempo. Quanto às desvantagens, tem-se o baixo desempenho quanto há poucos dados sobre o usuário e a falta de relacionamento entre os usuários (AGUIAR; FECHINE; COSTA, 2018; DA CONCEIÇÃO, et al., 2016).

2.1.1.2 Filtragem Colaborativa

Segundo Barros (2014) a técnica de filtragem colaborativa (FC) baseia-se na similaridade entre perfis de usuários, ou seja, como é visto na Figura 4, o usuário alvo tem seu perfil similar ao dos usuários 1 e 3, pois ambos gostaram dos mesmos livros, suas preferências são comparadas e os mesmos conteúdos serão recomendados para ambos e para quem mais for compatível com seus perfis, gerando assim um grupo de usuários.

O usuário alvo, que receberá as recomendações, gostou dos livros X, Y e Z, assim como os usuários 1 e 3, que por isso são os usuários mais similares ao usuário alvo. Além de terem gostado dos mesmos livros do usuário alvo, os usuários 1 e 3 gostaram do livro A e B, sendo assim o sistema vai recomendar esses dois livros ao usuário alvo.

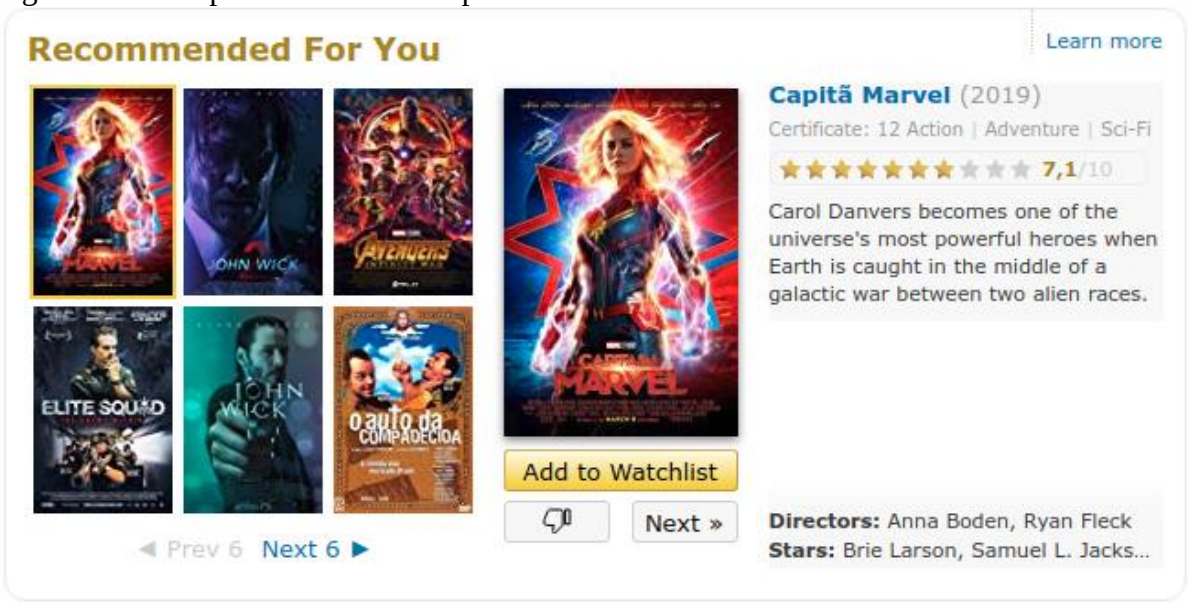
Figura 4 – Filtragem colaborativa



Fonte: Costa, Aguiar e Magalhães (2013).

Outra característica dessa técnica de filtragem é que os usuários podem avaliar os itens adquiridos através de pontuação. Essas pontuações são armazenadas na base de dados e grupos de pessoas com perfis similares são formados, assim outros usuários podem se beneficiar dessas pontuações (BARROS, 2014). Um exemplo de site que utiliza a filtragem colaborativa é o IMDb, uma plataforma muito popular onde se encontra informações sobre cinema e TV, permitindo que o usuário possa avaliar atribuindo uma nota para o que assistiu (SAMPAIO, 2006). Conforme demonstrado na Figura 5, os filmes são recomendados conforme o perfil de outras pessoas que deram notas altas para esses, e também deram uma boa nota para os mesmos filmes que o usuário avaliou.

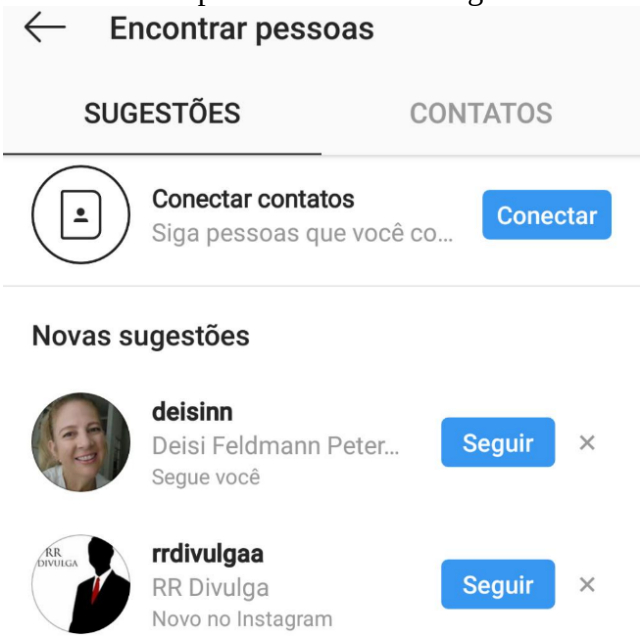
Figura 5 – Exemplo de FC utilizada por site IMDb



Fonte: IMDb (2019, texto digital).

Outro exemplo de filtragem colaborativa é a utilizada pela rede social Instagram, que recomenda aos usuários perfis de outros usuários que tenham gostos similares, localização próxima e seguidores em comum, conforme demonstrado na Figura 6.

Figura 6 – Exemplo de FC utilizada pela rede social Instagram



Fonte: Instagram (2019, texto digital).

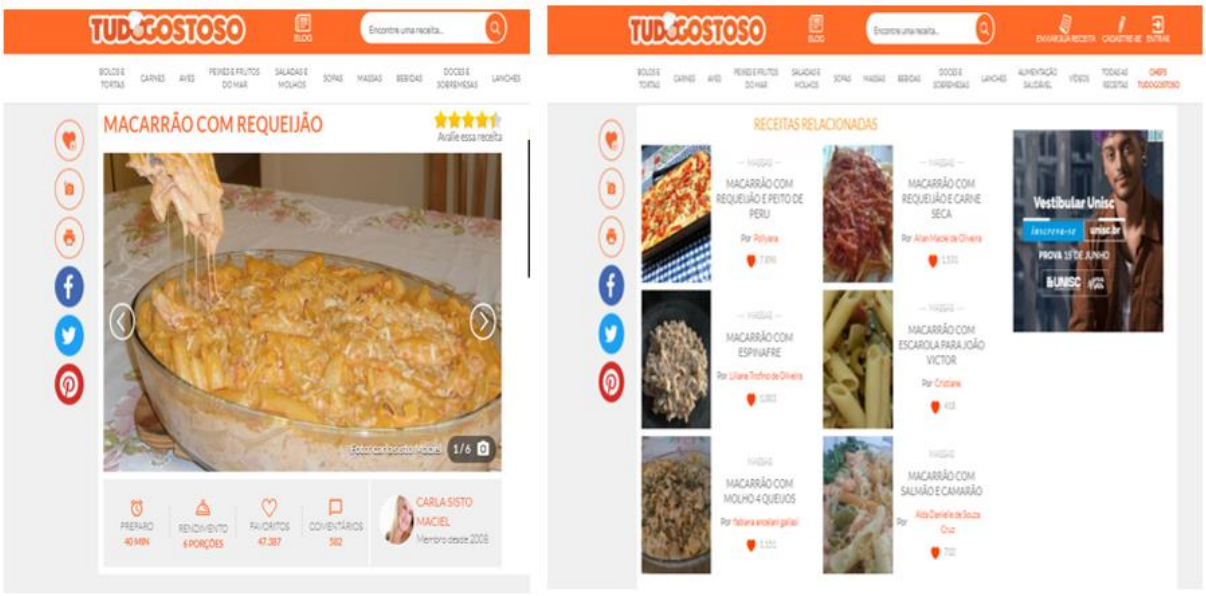
O processo de filtragem colaborativa acontece em três passos: representação dos dados de entrada, formação de vizinhança e geração da recomendação. Na etapa inicial de representação dos dados o usuário indica seus interesses e preferência, geralmente através de avaliações. Na etapa seguinte de formação de vizinhança, o sistema compara perfis de usuários para encontrar a similaridade e formar grupos de pessoas com interesses em comum. A etapa final é a geração da recomendação que filtra as avaliações feitas pelos componentes da vizinhança, o sistema gera então recomendações que visem agradar todos os usuários daquele grupo, conforme Mendes, Dos Santos e Picoli (2018). A maior vantagem da utilização da FC é a recomendação de itens com base no histórico de outros usuários relacionados. Já entre as desvantagens estão o baixo desempenho se o usuário não tiver uma quantidade considerável de relacionamentos e a impossibilidade de recomendar itens recém adicionados ao sistema que ainda tenham sido classificados por nenhum usuário (BARROS, 2014; MENDES; DOS SANTOS; PICOLI, 2018).

2.1.1.3 Filtragem Híbrida

A filtragem híbrida (FH) faz a combinação de duas ou mais técnicas, com o objetivo de aproveitar suas vantagens, de modo a desenvolver um sistema otimizado e que recomende o conteúdo mais adequado para o usuário (BARROS, 2014). A hibridização dessas técnicas possibilita a criação de um sistema capaz de ter bons resultados para usuários incomuns, recomendações precisas independentemente do número dos usuários, descoberta de similaridades entre os usuários e também a recomendação relacionada com o histórico do usuário (BARROS, 2014).

Um exemplo de filtragem híbrida pode ser visto no site de receitas Tudo Gostoso, onde o site recomenda e dá maior destaque as receitas melhores avaliadas pelos usuários e ao mesmo tempo faz a recomendação de receitas similares às que o usuário buscou. Este exemplo pode ser visto através da Figura 7.

Figura 7 – Exemplo de FH utilizada pelo site de receitas Tudo Gosto



Fonte: Tudo Gostoso (2019, texto digital).

2.1.2 Ferramentas de Filtragem

Para que se possa realizar a filtragem, independente da técnica escolhida, se fazem necessárias ferramentas sofisticadas que permitam que a mineração de dados seja feita com alto nível de eficiência e ao mesmo tempo de maneira fácil e prática. Um detalhamento acerca destas ferramentas é apresentado a seguir.

2.1.2.1 Aprendizagem por Máquina

Um sistema de recomendação exige uma programação sofisticada e de alto nível. Devido crescente demanda de sistemas deste tipo, a aprendizagem por máquina vem ganhando espaço nas pesquisas relacionadas à computação (AGGARWAL, 2016). A aprendizagem por máquina é um campo da inteligência artificial que torna os computadores capazes de aprender sem serem explicitamente programados. O processo de aprendizagem por máquina proporciona uma melhora em resultados gerados por computadores através da utilização de suas experiências anteriores como base. Esse tipo de ferramenta é utilizada para facilitar a construção de aplicativos, sites e sistemas inteligentes (QUILICI-

GONZALEZ; ZAMPIROLLI, 2015; AMARAL, 2016; FERRARI; SILVA, 2016). Apesar da aprendizagem por máquina ser uma tecnologia relativamente nova grandes empresas líderes do setor tecnológico já vêm explorando seus recursos, como é o caso do Facebook, Google, Netflix, Amazon e IBM.

O que torna a aprendizagem por máquina tão relevante é que ela abrange desde a análise da bolsa de valores até a construção de sistemas de recomendação de grandes sites de vendas, que recomendam produtos aos usuários com base em compras passadas e artigos similares. Outro grande campo de aplicação está na categorização de páginas de Web automaticamente conforme o tipo de conteúdo (moda, beleza, esportes, notícias, etc.). Um exemplo prático é o Gmail, que separa os e-mails por categoria de acordo com o conteúdo das mensagens, organizando os e-mails em principal, social, promoções, atualizações e fóruns. As técnicas de aprendizagem por máquina são utilizadas para solucionar problemas, as duas mais utilizadas são a aprendizagem supervisionada e não supervisionada (FERRARI; SILVA, 2016). As técnicas de aprendizagem supervisionada e não supervisionada serão detalhadas no texto a seguir.

2.1.2.1.1 Aprendizagem Supervisionada

A aprendizagem supervisionada tem como base a tarefa de aprender uma função a partir de dados de treinamento rotulados para prever o valor de qualquer entrada válida. Nela existe uma espécie de professor externo, que possuí um conhecimento sobre o assunto. Sendo assim, o algoritmo é treinado se baseando nos conhecimentos passados por seu professor (LORENA; CARVALHO, 2003; AMARAL, 2016; FERRARI; SILVA, 2016).

Em outras palavras a tarefa de atribuir sentido a dados é feita com base em exemplos do que é correto ou incorreto. Alguns exemplos comuns de aprendizagem supervisionada já foram apresentados na seção anterior, são eles: classificação de mensagens de e-mail, classificação de páginas de Web de acordo com seu conteúdo, entre outros. Os dois algoritmos mais utilizados na criação de aprendizes supervisionados são as Máquinas de Vetor de Suporte (SVMs) e os classificadores Naive Bayes (SANTOS; SANTOS, 2017).

O SVM é um método de aprendizagem de máquina que pode ser usado para problemas de classificação e regressão, dentre outras tarefas. Este método baseia-se na ideia de que vetores de entrada são não-linearmente mapeados para um espaço de atributos de alta dimensão. Nesse espaço, é construída uma superfície de decisão que permite distinguir as classes dos exemplos de entrada. o SVM tem como objetivo a construção de um hiperplano ótimo, que consiga diminuir a probabilidade de erro de classificação em relação ao conjunto de treinamento. Já o algoritmo de Naive Bayes consiste em um classificador probabilístico, baseado no Teorema de Bayes, que foi criado por Thomas Bayes, na tentativa de provar a existência de Deus. Atualmente é utilizado no aprendizado de máquina, principalmente na categorização de textos, baseando-se na frequência das palavras usadas (SANTOS; SANTOS, 2017).

Ele recebe o nome de naive (ingênuo) pois não considera a correlação entre as variáveis, tratando cada uma individualmente. Em outras palavras, o Naive Bayes utilizada a probabilidade condicional, por exemplo, qual a probabilidade de o evento A ocorrer, dado o evento B (FERRARI; SILVA, 2016).

2.1.2.1.2 Aprendizagem Não-Supervisionada

Na aprendizagem não supervisionada, a tarefa de atribuir sentido a dados é feita sem quaisquer exemplos do que é correto ou incorreto, sendo mais utilizada para agrupar entrada similar em grupos lógicos. Neste caso não existe a presença de um professor, ou seja, não existe um especialista que passará seus conhecimentos, o algoritmo aprende a interpretar as entradas segundo uma medida de qualidade (LORENA; CARVALHO, 2003; AMARAL, 2016; FERRARI; SILVA, 2016). Dentre as abordagens mais comuns à aprendizagem não supervisionada pode-se citar os mapas auto-organizadores, cluster k-Médio e hierárquico.

O mapa auto-organizável consiste em um tipo de rede neural artificial, que é treinada para produzir uma representação do espaço de entrada das amostras de treinamento. O agrupamento k-Médio é um método de Clustering que busca separar um número de observações em grupos, onde cada observação pertence ao grupo mais próximo da média (AMARAL, 2016; FERRARI; SILVA, 2016).

Já os algoritmos hierárquicos particionam os dados sucessivamente, criando uma hierarquia de relacionamentos, possibilitando assim, uma fácil visualização da formação dos agrupamentos e o grau de semelhança entre eles (AMARAL, 2016; FERRARI; SILVA, 2016).

2.2 Trabalhos Relacionados

Para compor este item buscou-se na literatura trabalhos acadêmicos e pesquisas em que sistemas de recomendação foram utilizados de maneiras diversas, como na recomendação de produtos/itens/serviços, na solução de problemas de sobrecarga de informação e no ganho de praticidade para o usuário.

Bussato (2013) criou um Sistema Web de Eventos chamado de “O Que Tá Valendo?”. O sistema desenvolvido baseou-se nos dados de grandes portais para obter a agenda cultural de Porto Alegre e recomendar atividades interessantes aos usuários. Na arquitetura do sistema foi utilizado como base de funcionamento o *framework* Yii, um *framework open-source* para a linguagem PHP, na versão 1.1.13, também foi necessário o uso de um *eb crawling* para extrair informações de sites e adicionar novos eventos ao sistema. Para realização do cadastro e login no sistema foi utilizada autenticação com o Facebook, o que permitiu o uso dos dados do perfil da rede social para fazer os cadastros. Foi ainda utilizado um banco de dados para armazenamento de informações do sistema.

Rodrigues (2019), propôs a criação de um SR para percursos culturais denominado #IWASHERE. O trabalho centrou-se na organização de ferramentas que permitissem recomendar um trajeto novo ou pontos de interesse novos a um utilizador no sistema. Para isso, foram utilizados dois cenários diferentes, o primeiro fez a semelhança entre utilizadores que percorreram trajetos e o segundo fez a semelhança entre utilizadores que visitaram os pontos de interesse. Ao final da pesquisa. a autora pôde concluir que o uso da filtragem colaborativa se mostrou o mais adequado, sendo também o mais utilizado nos estudos referentes de SR. Como resultado obteve-se um sistema inteligente que otimiza o trajeto levando em consideração o máximo de tempo que o utilizador pode dispensar e que máximo de Pontos de Interesse (POIs) que ele poderá ver. É também definido a distância máxima que dois POIs devem ter entre si, não cansando o utilizador que faz o trajeto.

Souza Filho (2018) desenvolveu um sistema de recomendação de eventos culturais com audiodescrição voltado para pessoas cegas usando dispositivos móveis Android, através de algoritmos de filtragem colaborativa baseada em item e filtragem baseada em conteúdo. A arquitetura utilizada pelo autor foi um servidor HTTP Apache em sua versão 2.2.34, com linguagem PHP 5.2.17 e banco de dados MySQL versão 5.6.39-83.1 em um sistema operacional Linux. O autor gerou uma potencial base de dados de eventos do gênero, um estudo acerca dos algoritmos propostos e um experimento de usabilidade da aplicação. Porém, identificou a ausência de métricas de avaliação estatística para as recomendações geradas, dada a abordagem pessoal utilizada pelos usuários na escolha das notas para os eventos, e relatou que seus resultados foram inconsistentes.

Canhoto (2013) realizou um estudo de criação de um sistema de recomendação de de música fazendo uma comparação entre filtragem colaborativa e *Context Filtering* (com base no conhecimento de preferências contextuais parciais do utilizador). Para o desenvolvimento do sistema o autor utilizou para o armazenamento e gestão dos dados o Sistema de Gestão de Bases de Dados (SGBD) MySQL. Já na camada aplicacional foram utilizados dois servidores distintos: O Apache HTTP Server e o Apache Tomcat. O Apache HTTP Server, foi utilizado para suportar o funcionamento online de todo o sistema à exceção do módulo de recomendação, que foi executado pelo servidor de programas escrito em Java Apache Tomcat. Quanto aos resultados, foram atribuídos 386 ratings (notas de critério) a 132 músicas por 82 participantes (com uma média de 5 ratings atribuídos por participante). Ao final do estudo observou-se que no início da utilização do sistema, o algoritmo de recomendação por filtragem colaborativa apresentou consistentemente recomendações que resultaram numa maior satisfação do utilizador. No entanto, à medida que o sistema foi sendo mais utilizado, a filtragem baseada no contexto superou as recomendações da filtragem colaborativa.

Gonçalves (2016) desenvolveu um protótipo de Sistema de Recomendação para séries de TV por assinatura, onde são recomendadas as séries com base nas avaliações realizadas pelo usuário, por meio da técnica de filtragem colaborativa. O sistema calcula a similaridade dos usuários aplicando o coeficiente de Pearson para estabelecer as vizinhanças e utiliza a média ponderada para gerar as recomendações. O SR foi desenvolvido para plataforma *web*, utilizando o PHP como linguagem de programação e, para o banco de dados, foi utilizado a o Sistema Gerenciador de Banco de Dados (SGBD)

MySQL. Participaram da validação do protótipo 15 usuários, dos quais 93% consideraram que as recomendações recebidas foram adequadas.

Da Rosa Furlan et al. (2018), realizam o desenvolvimento de um sistema de recomendação de artigos acadêmicos do Google Acadêmico, com o uso de técnicas de recomendação baseada em conhecimento e colaborativa. O perfil do usuário foi traçado a partir da extração dos dados do seu Currículo Lattes. O sistema foi desenvolvido para plataforma w*eb*, fazendo uso de linguagens de programação como Hypertext Preprocessor PHP, HTML, CSS e Javascript, bem como a manipulação da linguagem eXtensible Markup Language (XML). O MySQL foi escolhido para armazenamento e recuperação de dados, e o layout das páginas foi facilitado pelo uso do *framework* Bootstrap. A avaliação do sistema foi feita de forma qualitativa; em geral, as pessoas que testaram o sistema acharam o cadastro diferenciado e as recomendações iniciais recebidas adequadas.

3 METODOLOGIA

A metodologia consiste no estudo das ferramentas necessárias para realizar produção de cunho científico, sendo assim ela é de suma importância para elaboração deste projeto. Neste capítulo são apresentadas as técnicas de programação e metodologias utilizadas para desenvolver o sistema de recomendações proposto.

3.1 Tipo de Pesquisa

O presente trabalho apresenta o desenvolvimento de um sistema de recomendações de eventos, utilizando o *framework* Apache Mahout. Visando cumprir este objetivo foi feita uma pesquisa exploratória dos temas relacionados. Uma pesquisa de caráter exploratório desenvolve-se através de três etapas: o desenvolvimento de uma hipótese, a construção de um referencial teórico consistente para aumentar o conhecimento sobre o assunto a ser estudado e por último a identificação do fato ou fenômeno (MARCONI; LAKATOS, 2010).

De modo a enriquecer o trabalho foi realizada uma busca na literatura acerca do tema, o que levou a necessidade do uso do método de pesquisas bibliográficas. Segundo Gil (2002), este método de pesquisa faz o uso de referenciadas em bibliografias a respeito do tema da pesquisa, para poder melhor fundamentar o trabalho científico.

Para poder validar o sistema proposto, foi desenvolvido um protótipo, sendo o mesmo validado por estudantes da instituição através de uma pesquisa quali-qualitativa. A

pesquisa quali-qualitativa é aquela que concentra-se na qualidade do assunto abordado e também em dados numéricos (PRODANOV; FREITAS, 2013).

3.2 Materiais e Métodos

Na presente seção, serão descritas as principais ferramentas e tecnologias que serão utilizadas no desenvolvimento deste projeto.

3.2.1 Apache

Para a camada da aplicação, foi utilizado o Apache HTTP Server, um dos servidores web mais popular do mundo, sendo escolhido por sua fácil integração com outras tecnologias, como a linguagem de programação escolhida para o desenvolvimento, o PHP. Ele é o responsável pelo funcionamento online da interface de fonte de dados. O Apache é um software de código aberto mantido pela fundação Apache Group, que consiste em um servidor compatível com o protocolo HTTP, responsável por controlar as transações entre cliente e servidor, que pode ser executado em diversas plataformas e sistemas operacionais (APACHE, 2019).

3.2.2 Apache Mahout

Para o motor de recomendações foi utilizado o *framework* Apache Mahout, sua escolha foi dada devido ao fácil acesso a documentação da ferramenta, auxiliando no estudo desta tecnologia. O Apache Mahout é uma biblioteca de software livre, com algoritmos para mineração de dados distribuída, desenvolvida em Java e gerenciada pela Apache Software Foundation, cujo objetivo é criar algoritmos de aprendizagem por máquina (INGERSOLL, 2009).

Embora o Apache Mahout seja um projeto aberto a implementações de todos os tipos de técnicas de aprendizado de máquina, na prática é um projeto que se concentra em três áreas principais do aprendizado de máquina no momento. Esses são mecanismos de

recomendação (filtragem colaborativa), armazenamento em cluster e classificação (OWEN, 2012).

Ele fornece vários blocos de construção, onde podem ser criadas recomendações sobre o interesse de outras pessoas que tiveram interesse em comum em algum item, este tipo de recomendação é chamada de recomendação baseada em conteúdo, ou também comparar perfis de usuários, buscando usuários com gostos parecidos para criar recomendações interessantes para o usuário (MAHOUT, 2019).

3.2.3 PHP

A linguagem de programação PHP foi utilizada para o desenvolvimento do sistema responsável por integrar a aplicação que exibirá as recomendações e o Apache Mahout, sua escolha se dá pela experiência do acadêmico no trabalho e a fácil integração com outros sistemas da Univates.

O PHP é uma linguagem de programação *back-end* (processada pelo servidor), criada principalmente para o desenvolvimento web. O PHP será responsável por receber as requisições feitas no *front-end*, fazer o processamento destas requisições e apresentar um resultado, que pode ser retornar informações para a tela ou guardar registros no banco de dados (PHP, 2019).

3.2.4 Adianti *Framework*

Para auxiliar no desenvolvimento da interface de gestão de fontes de dados, foi utilizado o *framework* para PHP Adianti, sua escolha foi dada pela experiência do acadêmico no trabalho, o fácil acesso a documentação e a praticidade do uso.

O Adianti é um *framework* para a linguagem PHP *open-source*, baseado no padrão *Model, View, Controller* (MVC). Este padrão busca uma separação das camadas lógicas da aplicação, deixando o código gerado pelo desenvolvedor muito mais organizado e estruturado, além de oferecer componentes de alto nível para criação de aplicações (ADIANTI, 2019).

3.2.5 PostgreSQL

Como Sistema de Gestão de Bases de Dados (SGBD) foi utilizado o PostgreSQL. A escolha foi feita em virtude da experiência do acadêmico em outros projetos com a ferramenta, da documentação de fácil acesso e da fácil integração com outras tecnologias utilizadas.

É um sistema de gerência de banco de dados relacional de código aberto, que usa e amplia a linguagem *Structured Query Language* (SQL), combinando-a com diversos recursos para armazenamento e dimensionamento de dados. É chamado de relacional pois os dados são organizados em forma de tabelas, ou seja, linhas e colunas relacionadas através de chaves estrangeiras (POSTGRESQL, 2019).

4 ESPECIFICAÇÕES DO PROJETO

O objetivo do presente capítulo é apresentar o projeto proposto, desde sua visão geral, definição de seus requisitos, modelo de casos de uso e modelo de estados.

4.1 Visão Geral do Projeto

O objetivo geral deste projeto é criar um sistema de recomendações de eventos utilizando a biblioteca Apache Mahout. A solução apresentada neste trabalho envolve a modelagem e implementação de um protótipo de Sistema de Recomendação de serviços oferecidos pela Univates, tais como cursos, palestras e shows. Pretende-se, por meio da filtragem colaborativa, recomendar os eventos mais adequados para cada usuário.

O sistema de recomendações, buscará informações nos sistemas de inscrições da Univates, onde são oferecidos diversos tipos de eventos, tais como, palestras, show e amostras de trabalhos acadêmicos, e no sistema de cursos de extensão, onde são oferecidos cursos para a comunidade, para assim traçar o perfil de cada usuário e compara-lo com os demais, para melhor divulgação aos possíveis interessados.

Para a comunicação entre a interface que exibirá as recomendações e o base de dados com as informações dos usuários, será desenvolvida uma *Application Programming Interface* (API), que será a responsável por tratar os dados dos usuários, formatá-los e enviar para o Apache Mahout gerar as recomendações, que por sua vez, deve encaminhar as recomendações de volta a API, culminando na exibição das mesmas na interface do usuário.

A API será chamada através de um webservice Simple Object Access Protocol (SOAP), recebendo o identificador do usuário que receberá as recomendações e o tipo de serviço que será recomendado. Cada tipo de serviço terá um classe dentro da API, assim, caso seja disponibilizado um novo serviço pela instituição, basta criar uma nova classe para permitir que o mesmo também possa obter recomendações. Estas classes serão as responsáveis por buscar os dados na base e formatá-los para o modelo de dados do Apache Mahout, além de devolver as recomendações geradas para o sistema que fez a solicitação.

Na Figura 8, está representado o fluxo da proposta, onde usuários que utilizam os sistemas geram informações, inscrições, matrículas, interesses em eventos, que são gravados no banco de dados. Ao mesmo tempo em que diferentes usuário estão abastecendo esse banco, o Apache Mahout recebe esses dados e busca encontrar perfis semelhantes ao da pessoa que está acessando o sistema, para assim fazer as recomendações que serão exibidas na tela.

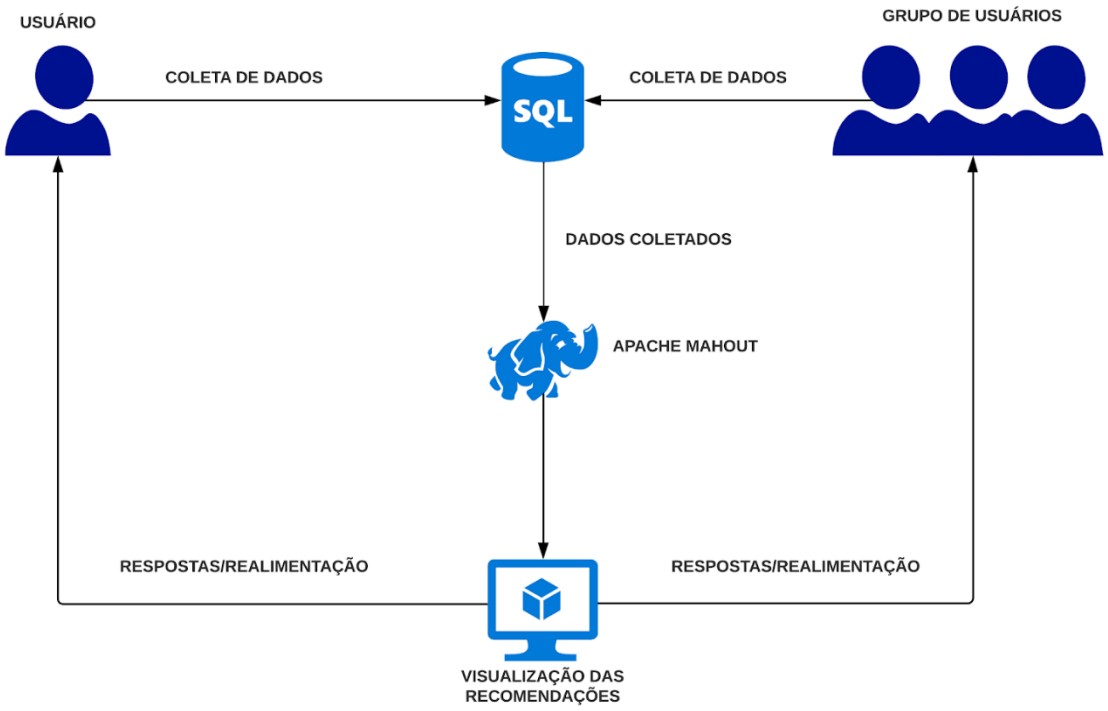
Para melhor entendimento, o fluxo da aplicação está dividido em três partes, sendo elas:

a) Usuário: Usuário que acessa os ambientes virtuais da Univates; possível interessado em algum serviço;

b) Banco de dados: Banco de dados onde estão os registros dos outros usuários, ou seja, sua vizinhança;

c) Apache Mahout: Com base na vizinhança e no usuário, vai fazer os cruzamentos dos dados, buscando similaridades, para assim fazer as recomendações.

Figura 8 – Fluxograma do projeto



Fonte: Do Autor (2019).

Para fazer as recomendações baseadas em usuário, o Apache Mahout disponibiliza a interface UserSimilarity, também é preciso indicar a vizinhança, ou seja, quantos usuários similares serão utilizados para encontrar itens para recomendar, que é representada pela interface UserNeighborhood. Para o Apache Mahout processar os dados, eles vão estar em arquivo csv gerado pela API, onde estão informados o id do usuário, o id do item, separados por vírgula, conforme mostra a Figura 9.

Figura 9 – Modelo de dados no Apache Mahout

1,	101
1,	102
1,	103
2,	101
2,	102
2,	103
2,	104
3,	101
3,	104
3,	103
4,	105
4,	102
4,	101
5,	102
5,	103
5,	105

Fonte: Do Autor (2019).

4.2 Requisitos

Existem basicamente dois tipos de requisitos, os funcionais, que são funcionalidades ou serviços que o sistema deve ter, e os não funcionais, que são aqueles que definem como as funcionalidades serão implementadas.

4.2.1 Requisitos Funcionais

Para o desenvolver o sistema proposto, alguns requisitos funcionais precisam ser atendidos. Na Tabela 1, serão elencados o principal grupo de requisitos funcionais da aplicação:

Tabela 1 – Requisitos funcionais

RF01 - Gerar Recomendações aos usuários	**Prioridade**
Gerar as recomendações dos eventos realizados pela univates aos usuários, baseado na similaridade do seu histórico de inscrições de eventos com os demais usuários da Universidade do Vale do Taquari Univates	Alta
RF02 - Exibir as Recomendações	**Prioridade**
Entregar as recomendações que foram geradas ao usuário	Alta
RF03 - Desenvolver API para integração entre sistemas	**Prioridade**
Desenvolver API de comunicação da plataforma, ela será responsável por receber os dados do sistema de inscrições e montar o arquivo para o Apache Mahout, e também receber o arquivo de recomendações do Apache Mahout, e devolver o objeto para o sistema que fez a requisição.	Alta
RF04 - Alertas de erros em serviços	**Prioridade**
Tela para visualização de logs caso ocorra algum erro na aplicação, ou no formato do arquivo.	Média
RF05 - Cadastro de usuários	**Prioridade**
Cadastrar os usuários que terão acesso ao protótipo, onde serão exibidas as recomendações.	Alta
RF06 - Controle de Acessos	**Prioridade**
Controle para permissão de acesso ao protótipo	Alta

Fonte: Do Autor (2019).

4.2.2 Requisitos Não-Funcionais

A seguir serão apresentados os principais requisitos não funcionais serão implementados para o projeto proposto, conforme Tabela 2.

Tabela 2 – Requisitos não-funcionais

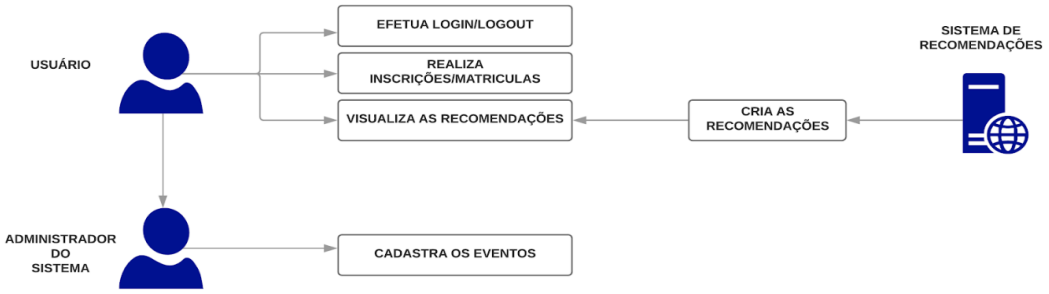
RNF01 - Utilizar um servidor Apache	**Prioridade**
Utilizar o servidor HTTP Apache para rodar a aplicação	Alta
RNF02 - Utilizar o *FrameWork* Apache Mahout	**Prioridade**
Será o responsável por realizar os cruzamentos de usuários e retornar as recomendações.	Alta
RNF03 - Utilizar HTML5, CSS3	**Prioridade**
Desenvolver a interface front-end seguindo com padrões HTML5 e CSS3	Alta
RNF04 - Utilizar PHP 7.2	**Prioridade**
Desenvolver utilizando PHP 7.2 como linguagem de programação back-end	Alta
RNF05 - Utilizar Adianti *Framework*	**Prioridade**
Utilizar Adianti *Framework* no desenvolvimento do *back-end* da api que fará a ligação entre o Apache Mahout e a aplicação que exibirá a recomendação.	Alta
RNF06 - Utilizar PostgreSQL	**Prioridade**
Utilizar como banco de dados da aplicação o PostgreSQL versão 11	Alta
RNF07 - Compatível com navegador Google Chrome	**Prioridade**
Ser compatível com navegadores Google Chrome (versão 70 ou superior)	Alta

Fonte: Do Autor (2019).

4.3 Modelo de Casos de Uso

Para demonstrar os principais papéis dentro do projeto proposto, será utilizado um diagrama de casos de uso. Como visto na Figura 10. São dois grupos de usuários, os que cadastram os eventos a serem oferecidos, e o público alvo, que vai fazer a sua inscrição. Os administradores são responsáveis por cadastrar os eventos nos sistemas, informando os dados necessários para disponibilizá-los para o público, tais como o tipo do evento, as datas de inscrição e valores. Já o usuário trata-se do público, é quem utiliza o sistema buscando eventos para se inscrever e recebe as recomendações.

Figura 10 – Modelo de casos de uso

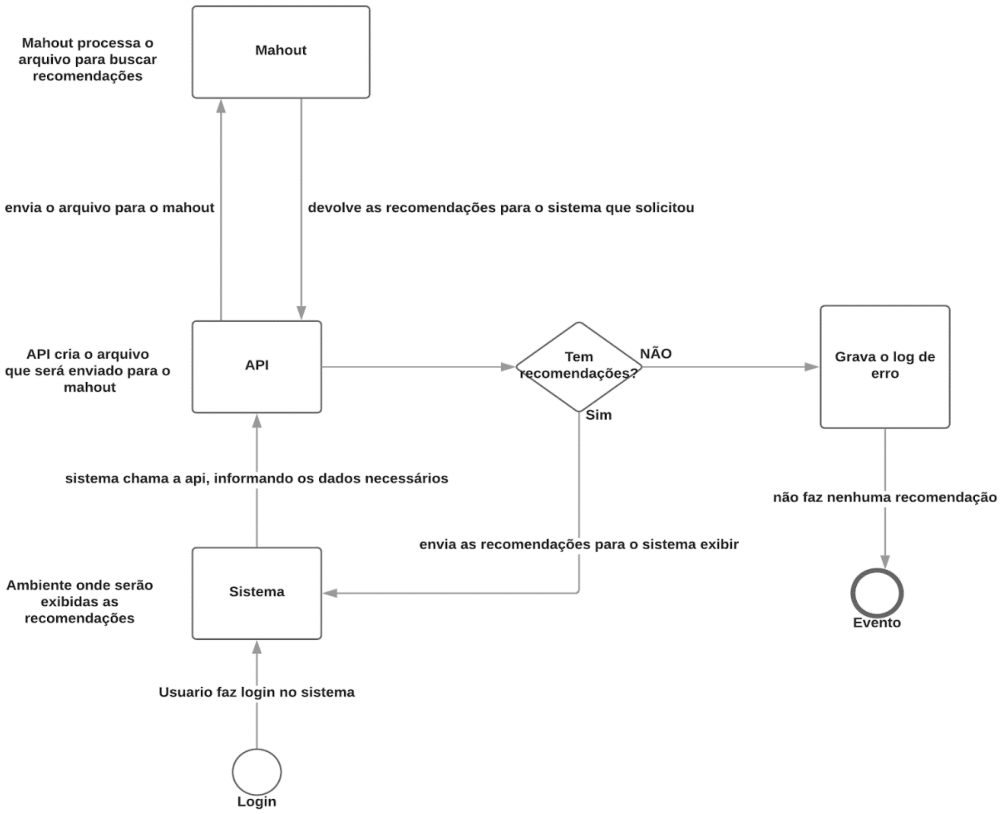


Fonte: Do Autor (2019).

4.4 Diagrama de Estados

A Figura 11 representa um diagrama de estados, buscando reproduzir o cenário, onde um usuário faz o login em algum dos sistemas onde serão exibidas recomendações. Este sistema chama a API, passando para ela os dados necessários para a API criar o arquivo que será enviado para o Apache Mahout realizar as recomendações. A API recebe o retorno do Apache Mahout e o processa, caso existam recomendações, ele as retorna para o sistema que fez a solicitação.

Figura 11 – Diagrama de estados



Fonte: Do Autor (2019).

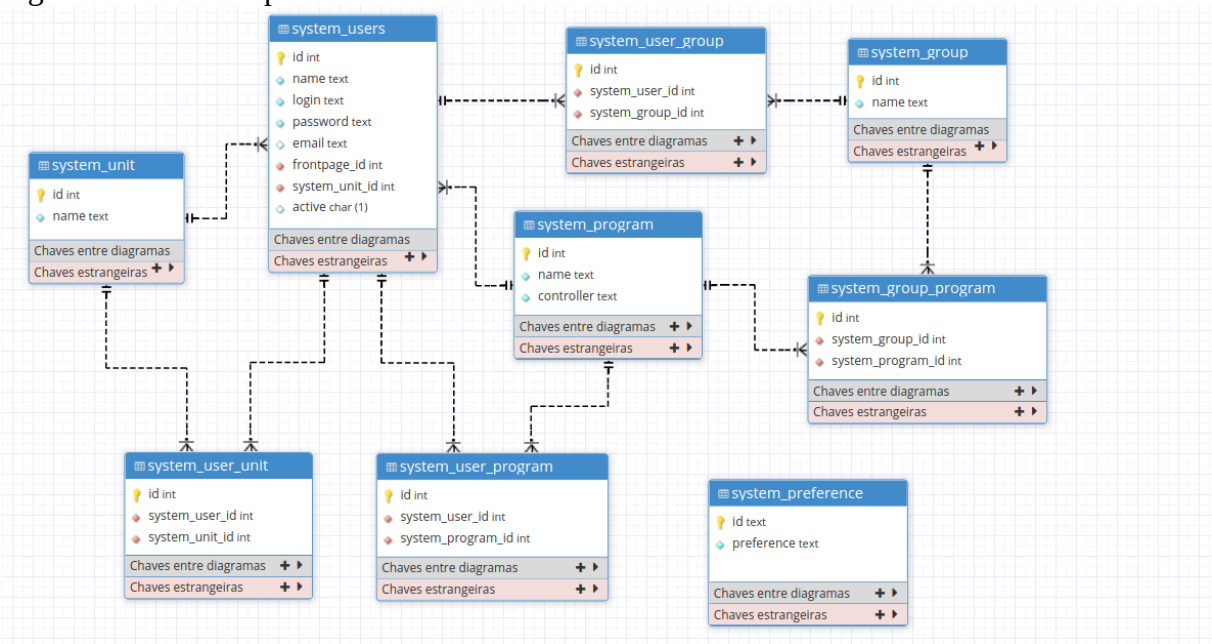
5 IMPLEMENTAÇÃO

Neste capítulo é apresentada a implementação do projeto, especificando tecnologias e algoritmos utilizados.

5.1 Estrutura do Banco de Dados

O banco de dados do protótipo foi estruturado em duas partes, a primeira contempla a estrutura obrigatória para utilização do controle de permissões de acesso, disponibilizada pelo *framework* Adianti, contendo cadastros completos de usuários, grupos, unidades, programas e permissões. A Figura 12 mostra o modelo ER desta estrutura.

Figura 12 – Tabelas para controle de acesso do *Framework*



Fonte: Adianti (2019, texto digital).

A Tabela 3 especifica a função de cada tabela da estrutura para controle de acessos disponibilizada pelo *Framework* Adianti.

Tabela 3 – Especificação da estrutura de controle de acesso do *Framework* Adianti

Elemento	Função
system_users	Representa um usuário do sistema.
system_unit	Representa uma unidade da organização, pode ser uma filial, um departamento, ou outro tipo de divisão. Um usuário estará vinculado à uma unidade.
system_group	Representa um grupo de usuários. Um usuário poderá fazer parte de muitos grupos.
system_program	Representa um programa. Basicamente é uma página representada por uma classe em app/control.
system_user_group	Vincula usuários com grupos.
system_user_program	Vincula usuários com programas.
system_group_program	Vincula grupos de usuários com programas.
system_preference	Armazena as preferências gerais do sistema.
system_user_unit	Armazena as unidades do usuário.

Fonte: Do Autor (2019).

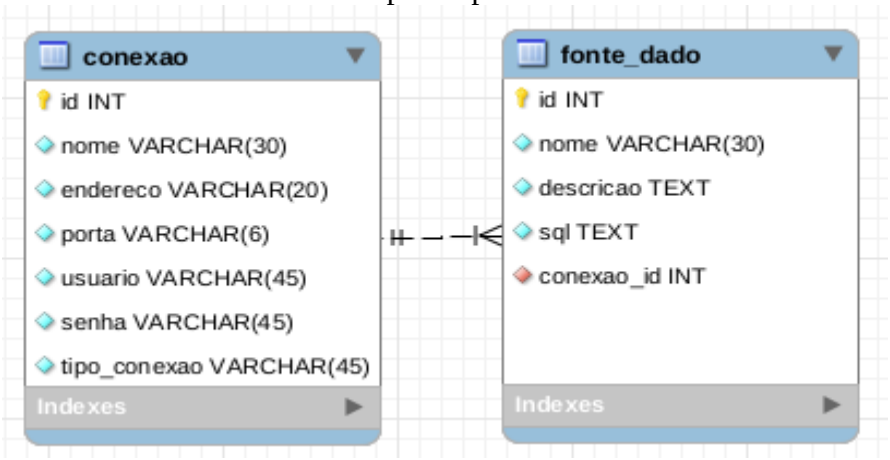
Na segunda parte do banco de dados são armazenados os recursos necessários para gerar as recomendações, sendo composta por duas tabelas: “conexao” e “fonte_dado”.

Na tabela “conexao”, são cadastrados os sistemas de onde são buscados os dados para a formação do DataModel que será utilizado para gerar as recomendações. Nesta tabela estão presentes os atributos “id” que é a chave primária da tabela, “nome”, “endereco” representando o endereço IP do servidor, “porta” para porta de conexão, “usuario” e “senha” que terão permissão de conexão, e o atributo “tipo_conexao” que armazena o tipo da conexão que será utilizado, que pode variar dependendo do modelo do banco de dados utilizado pelo sistema que fornecerá os dados.

Na tabela “fonte_dado”, são inseridos as consultas que buscam os dados no banco de dados das conexões cadastradas anteriormente. São atributos desta tabela um “id” que é a chave primária da tabela, o atributo “nome” para ajudar a identificar a fonte de dados, o

campo “descricao” para informar uma descrição que ajude a facilitar o entendimento do que a consulta vai retornar, um campo “conexao_id” que faz referência a tabela “conexao” e o campo “sql”, onde é armazenada a consulta em linguagem SQL que será executada para retornar os dados que alimentarão o DataModel. A Figura 13 mostra o modelo da estrutura do banco de dados do protótipo.

Figura 13 – Modelo de banco de dados do protótipo



Fonte: Do Autor (2019).

5.2 Implementação do Sistema de Recomendação

O sistema de recomendações foi desenvolvido fazendo uso do *framework* Apache Mahout. O sistema de recomendações recebe como parâmetro o código da pessoa que receberá as recomendações, e o nome do arquivo utilizado pelo DataModel. O DataModel é onde estão as informações das preferências do usuário. Estes dados podem estar tanto em um arquivo quanto em uma base de dados (OWEN, 2012).

O Apache Mahout fornece a classe MySQLJDBCDataModel para integrações para JDBC e MySQL, e também uma implementação do JDBCDataModel para PostgreSQL ou até uma classe GenericJDBCData-Model que permite usar dados de bancos de dados que não têm implementação especializada, para aplicações que vão buscar os dados diretamente na fonte de dados. Também fornece um FileDataModel, onde os dados estão em um arquivo .csv (Owen, 2012).

No DataModel, usuários e itens são identificados apenas por um valor numérico. Um terceiro campo, com a força da preferência do item pode ser informado, que pode ser qualquer

número, desde que valores maiores representem preferências positivas mais fortes. Caso não tenha valores de preferência, o Apache Mahout suporta o modelo de dados "booleano", no qual os usuários não expressam preferências, ou seja existe apenas um noção de associação, ou nenhuma, entre um usuário e os itens. (OWEN, 2012). No caso do presente trabalho, o arquivo é composto pelo código que representa a pessoa dentro dos sistemas da Univates e o código do serviço que a pessoa realizou a inscrição. Os serviços representados dentro da Datamodel podem variar de acordo com o que se deseja recomendar, eles podem representar cursos, eventos, palestras ou algum outro serviço.

A interface UserSimilarity é onde são disponibilizadas classes que fazem os cálculos para a noção de similaridade entre dois usuários, sendo assim parte crucial de um mecanismo de recomendação (OWEN, 2012).

A UserSimilarity é a interface que define a implementação das classes que fazem os cálculos para a noção de similaridade entre dois usuários, sendo assim parte crucial de um mecanismo de recomendação. As principais implementações da interface UserSimilarity, quando se tem conhecimento de valores de força preferência, são as classes PearsonCorrelationSimilarity e a EuclideanDistanceSimilarity. Caso não haja valor de força da preferência, existem as classes LogLikelihoodSimilarity e TanimotoCoefficientSimilarity (OWEN, 2012).

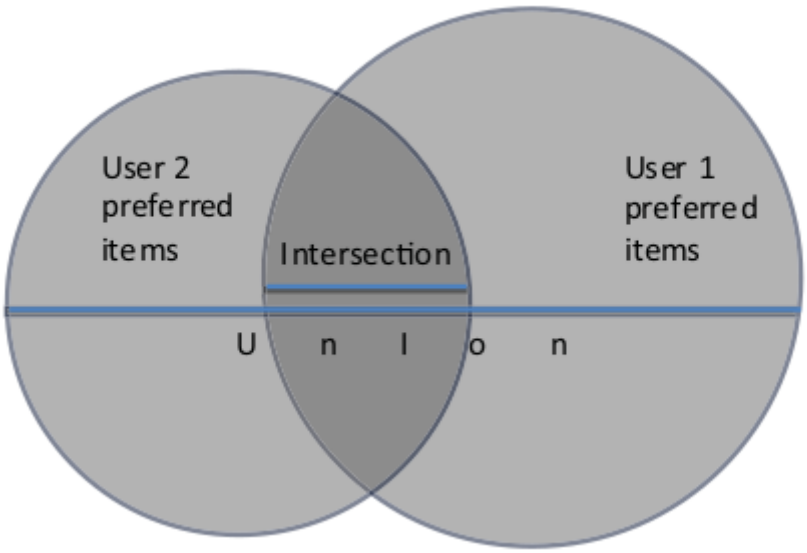
Como a Univates não utiliza nenhum sistema para que os usuários possam dar uma nota para expressar sua satisfação em relação a um evento após sua participação, não se tem noção da força das associações entre usuários e eventos. Na linguagem do Apache Mahout, essas associações sem valores de preferência são chamadas de preferências booleanas porque uma associação pode ter um de dois valores: ela existe ou não existe (OWEN, 2012). Desta forma, foram utilizadas para calcular a métrica de semelhança, as classes LogLikelihoodSimilarity e TanimotoCoefficientSimilarity.

O TanimotoCoefficientSimilarity é uma implementação com base no coeficiente de Tanimoto; esse valor também é conhecido como coeficiente de Jaccard. Ele representa o número de itens que dois usuários escolheram em comum, dividido pelo número de itens que estes mesmos usuários escolheram, conforme ilustrado na Figura 14. Já o LogLikelihoodSimilarity faz um cálculo similar ao coeficiente de Tanimoto, porém ele avalia

também se não foi uma casualidade a preferência do usuário por determinado item (OWEN, 2012).

Esta interface está ligada diretamente a classe UserNeighborhood, pois através dos usuários semelhantes que será definida a “vizinhança” para o usuário (MAHOUT, 2019).

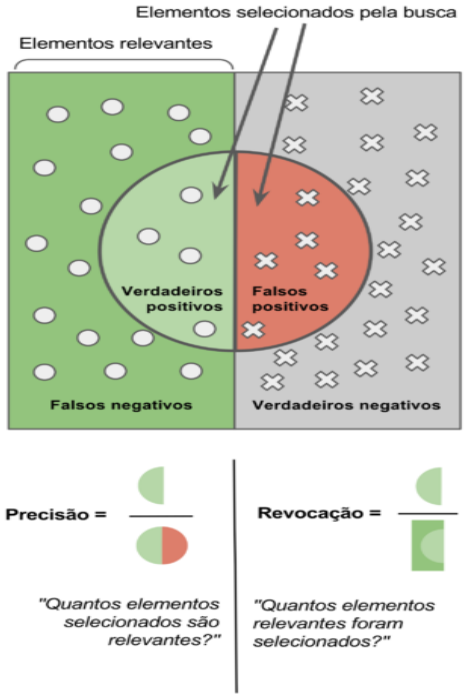
Figura 14 – Coeficiente de Tanimoto



Fonte: Owen (2012).

O Apache Mahout fornece formas de avaliar o desempenho do recomendador criado. Dentre as métricas disponíveis, podemos citar a possibilidade de obter os valores de precisão e *recall*, utilizando a classe GenericRecommenderIRStatsEvaluator. A Precisão representa a relação entre o que o sistema recomendou e o que o usuário realmente teve interesse. Já o *recall* é a proporção de boas recomendações que aparecem no topo da lista de recomendações (OWEN, 2012). A Figura 15 ilustra a diferença entre precisão e *recall*.

Figura 15 – Precisão e *recall*



Fonte: Wikipedia (2019, texto digital)

O teste de precisão e *recall* foi foi utilizado para auxiliar a encontrar o melhor algoritmo para métrica de semelhança entre LogLikelihoodSimilarity e TanimotoCoefficientSimilarity, além de encontrar o melhor número para o tamanho da vizinhança que será utilizada, foram feitos testes com vizinhanças de dez, vinte, trinta e quarenta usuários.

Neste teste, para cada usuário o Apache Mahout remove as N principais preferências da Datamodel, onde N é informado no quinto parâmetro na função *evaluate*, que refaz o cálculo de recomendações para as novas preferências, comparando assim quanto dos novos itens recomendados estão dentro das reais preferências do usuário; cada teste durou cerca de 2 horas.

Para calcular as novas preferências de item para um usuário, precisamos considerar as preferências de usuários semelhantes. Um conjunto de usuários semelhantes ao usuário atual é chamado de vizinhança. No Apache Mahout, a noção de vizinhança é definida pela interface UserNeighborhood, que possui as implementações NearestNUserNeighborhood e ThresholdUserNeighborhood. Basicamente, na classe NearestNUserNeighborhood é definido um número de usuários mais semelhantes que serão utilizados para gerar recomendações, já no ThresholdUserNeighborhood não é definido o número usuários semelhantes, mas sim o

valor mínimo de semelhança com usuário-alvo para poder pertencer a “vizinhança” (OWEN, 2012; MAHOUT, 2019).

Para a criação da vizinhança no protótipo, utilizou-se a classe NearestNUserNeighborhood, pois assim é possível obter recomendações mesmo que a vizinhança não tenha uma semelhança muito elevada. Caso fosse utilizada a ThresholdUserNeighborhood, que se baseia em um limite de semelhança, haveria o risco de não encontrar uma vizinhança adequada, com isso não gerando recomendações.

Conforme pode ser visto na Figura 16, para este teste foi determinado o número N mencionado acima, foi de 10, além disso é necessário definir um limite (threshold) para determinar quando a recomendação é de fato boa, para isso foi utilizada a propriedade GenericRecommenderIRStatsEvaluator.CHOOSE_THRESHOLD que define automaticamente esse valor.

Figura 16 – Teste de precisão e *recall*

```
RandomUtils.useTestSeed();
DataModel model = new FileDataModel(new File("inscricoes.csv"));
RecommenderIRStatsEvaluator evaluator = new GenericRecommenderIRStatsEvaluator();

RecommenderBuilder recommenderBuilder = new RecommenderBuilder()
{
    public Recommender buildRecommender(DataModel model) throws TasteException
    {
        UserSimilarity similarity = new LogLikelihoodSimilarity(model);
        UserNeighborhood neighborhood = new NearestNUserNeighborhood(10, similarity, model);
        return new GenericBooleanPrefUserBasedRecommender(model, neighborhood, similarity);
    }
};
DataModelBuilder modelBuilder = new DataModelBuilder()
{
    public DataModel buildDataModel(FastByIDMap<PreferenceArray> trainingData)
    {
        return new GenericBooleanPrefDataModel(GenericBooleanPrefDataModel.toDataMap(trainingData));
    }

};

IRStatistics stats = evaluator.evaluate(recommenderBuilder,
                                        modelBuilder,
                                        model,
                                        null,
                                        10,
                                        GenericRecommenderIRStatsEvaluator.CHOOSE_THRESHOLD,
                                        1.0);


System.out.println("limite" + GenericRecommenderIRStatsEvaluator.CHOOSE_THRESHOLD);
System.out.println("precisão" + stats.getPrecision());
System.out.println("recall" + stats.getRecall());



}
```

Fonte: Do Autor (2019).

O melhor resultado obtido foi utilizando o algoritmo de similaridade "LogLikelihoodSimilarity" com uma vizinhança de vinte usuários, que alcançou precisão de 0,4311 (43,1%) e o recall de 0,4311 (43,1%). A Tabela 4 ilustra os resultados dos testes de precisão e recall, onde ambos podem variar entre 0 e 1.

Tabela 4 – Teste de precisão por *recall*

Algoritmo	Precision	Recall	Vizinhança (10)
LogLikelihood	0,4263	0,4260	10
LogLikelihood	0,4311	0,4311	20
LogLikelihood	0,4283	0,4282	30
LogLikelihood	0,4221	0,4220	40
Tanimoto	0,4144	0,4141	10
Tanimoto	0,4263	0,4262	20
Tanimoto	0,4288	0,4285	30
Tanimoto	0,4256	0,4254	40

Fonte: Do Autor (2019).

Após identificação do algoritmo de similaridade e parametrização mais adequada ao conjunto de dados utilizado, foi criado o motor de recomendação, onde o método *recommend*, da interface *Recommender* recebe dois parâmetros, o código do usuário que se destinam as recomendações e o número máximo de recomendações a serem retornadas.

O *Recommender* é a peça central, é nele que são centralizados os outros três componentes necessários para gerar as recomendações. As suas principais implementações são as classes GenericUserBasedRecommender e GenericItemBasedRecommender, para recomendações baseadas em usuários e baseadas em itens (MAHOUT, 2019)

Na Figura 17 é apresentada a implementação do sistema de recomendações.

Figura 17 – Sistema de recomendações

```
public static String[] recomendar( Long cod_pessoa, String dados ) throws IOException, TasteException
{
    DataModel model = new FileDataModel(new File(dados+".csv"));

    DataModel model_ativos = new FileDataModel(new File(dados+".csv"));
    UserSimilarity similarity = new LogLikelihoodSimilarity(model);
    UserNeighborhood neighborhood = new NearestNUserNeighborhood(10, similarity, model);

    GenericBooleanPrefUserBasedRecommender recommender = new GenericBooleanPrefUserBasedRecommender(model_ativos, neighborhood, similarity);

    List<RecommendedItem> recommendations = recommender.recommend(cod_pessoa, 5);

    String retorno[] = new String[10];
    int i = 0;
    for (RecommendedItem recommendation : recommendations)
    {

        int cod_item = (int) recommendation.getItemID();
        float valor  = recommendation.getValue();
        retorno[i] = cod_item+"-"+valor;
        i++;
    }

    return retorno;
```

Fonte: Do Autor (2019).

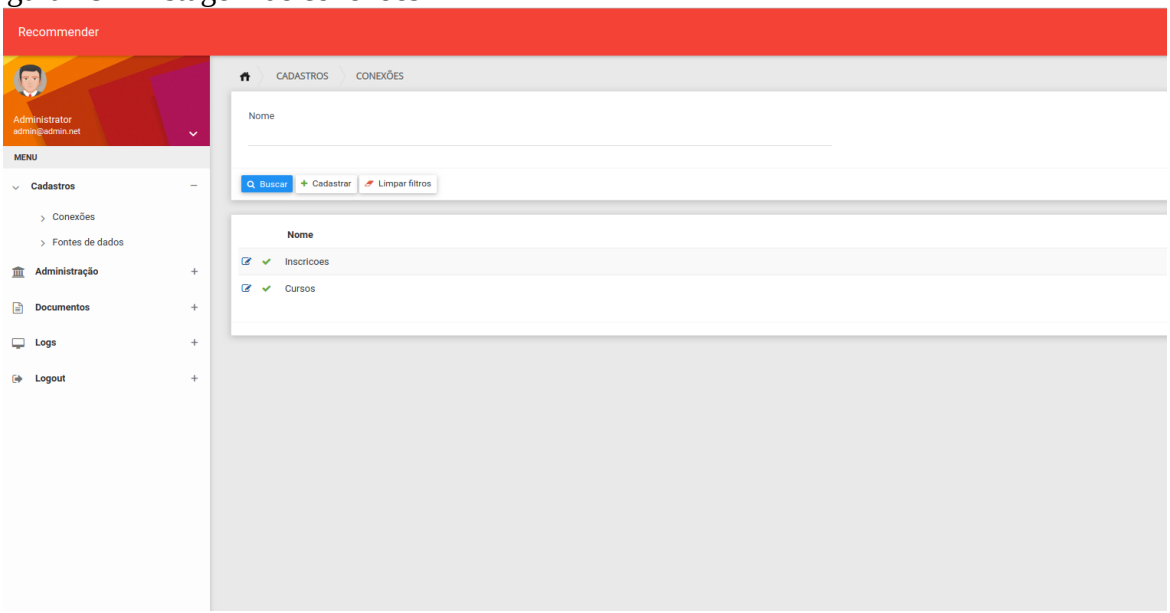
5.3 Implementação da Interface de Gestão de Fontes de Dados

A Interface de gestão de fontes de dados é a aplicação que faz o intermédio entre o sistema que exibirá as recomendações e o sistema de onde são buscadas as preferências dos usuários e os eventos que serão recomendados. Neste protótipo são cadastrados os as conexões e as fontes de dados para gerar as recomendações.

O protótipo implementado apresenta uma tela de login, possibilitando que somente pessoas com permissão de acesso possam logar nele.

Após efetuar login, pode-se realizar os cadastros das fontes de dados e conexões, para isso basta acessar o menu lateral, na seção “Cadastros” e escolher o cadastro que deseja fazer, caso seja escolhido o cadastro de “Conexões”, será exibida uma listagem com todas as conexões já cadastradas, onde é possível buscar uma conexão através do nome, editar as conexões já existentes, testar uma conexão ou criar uma nova conexão. A Figura 18 ilustra a tela de listagem de conexões.

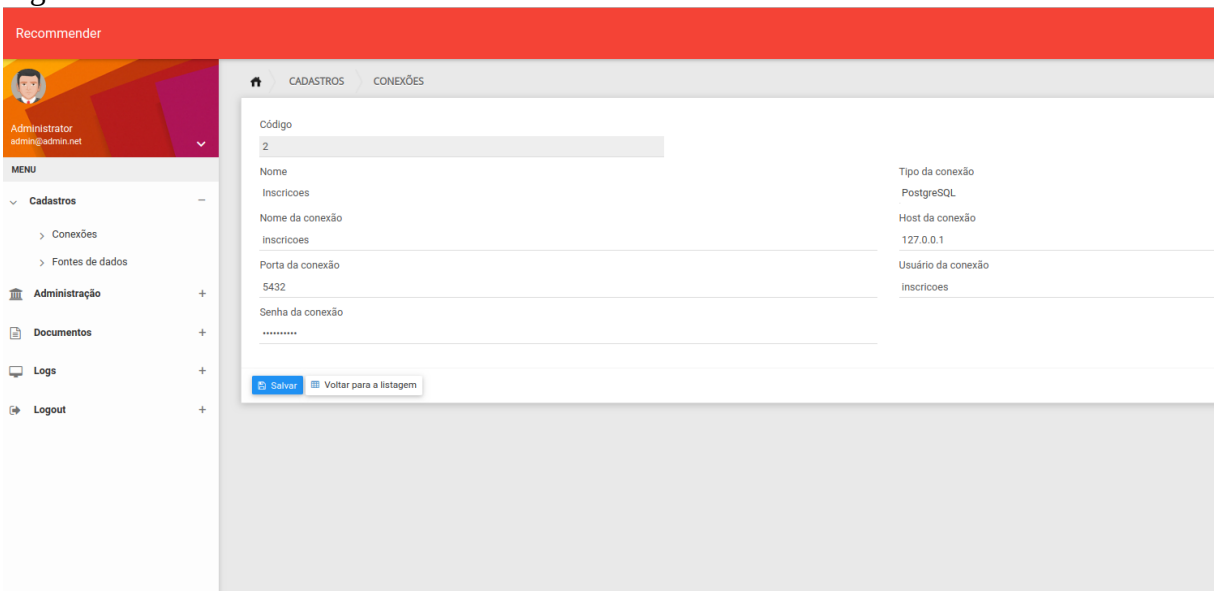
Figura 18 – Listagem de conexões



Fonte: Do Autor (2019).

Para cadastrar uma nova conexão, o usuário deve clicar no botão “Cadastrar” da listagem, ele será redirecionado para a tela de cadastro, nela ele deve informar os campos para criar uma nova conexão, conforme mostra a Figura 19.

Figura 19 – Cadastro de conexões



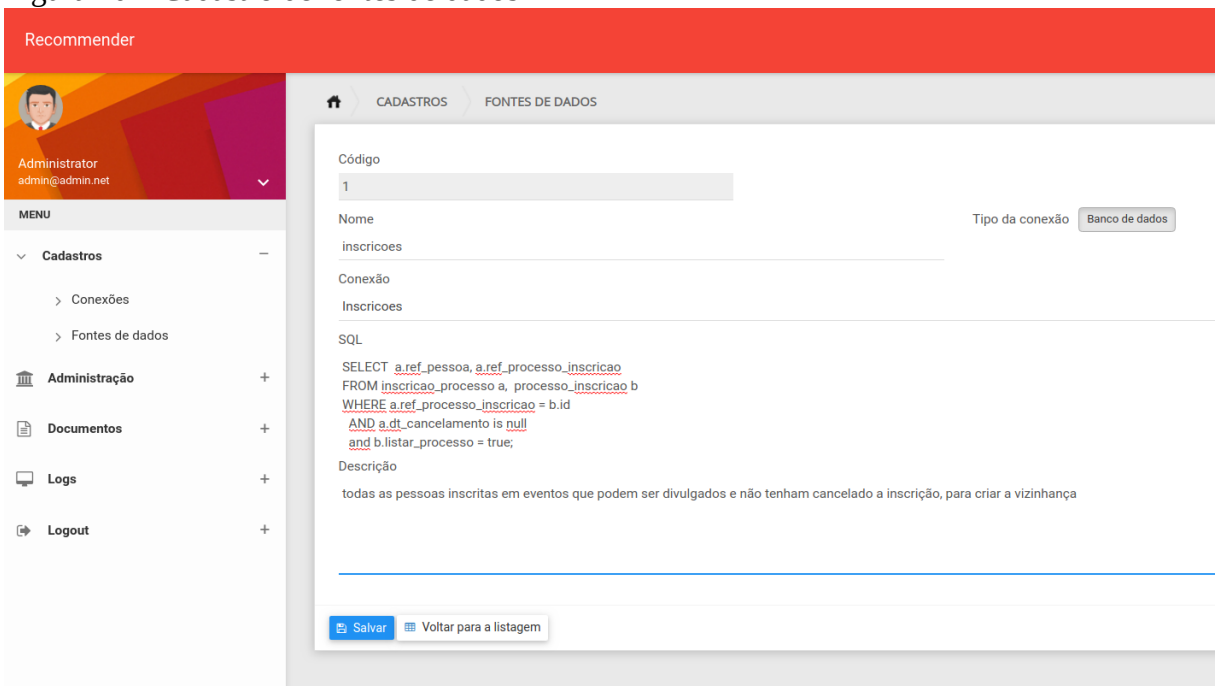
Fonte: Do Autor (2019).

Tendo criado as conexões, deve-se cadastrar a fonte de dado, pois é através dela que será realizada a consulta para criar os arquivos contendo as preferências dos usuários. Assim como no cadastro de conexões, o cadastro de fonte de dado também possui uma listagem,

onde é possível editar uma fonte de dados já cadastrada, criar o arquivo que será utilizado pelo DataModel no Apache Mahout e cadastrar uma nova fonte de dados.

A Figura 20 mostra a tela para cadastro de uma nova fonte de dados, onde o usuário cadastra uma consulta SQL que será utilizado para montar o arquivo CSV com os dados necessários para gerar as recomendações. Neste caso a fonte de dados cadastrada está buscando o código do usuário e o código do evento, na tabela de processos do sistema de inscrições, para isso, está utilizando a conexão cadastrada anteriormente chamada de "inscricoes", que permite realizar a consulta dentro do sistema de inscrições.

Figura 20 – Cadastro de fontes de dados



Fonte: Do Autor (2019).

Ainda dentro da Interface de gestão de fontes de dados, na classe "Recomendador" está a função "buscaRecomendacoes" ilustrada na Figura 21. Ela é a responsável por fazer a chamada do sistema de recomendações, informando qual a pessoa que receberá as recomendações e a fonte de dados que será utilizada para carregar o DataModel do sistema de recomendação. Esta chamada executa o recomendador desenvolvido no Apache Mahout para gerar recomendações baseadas no arquivo que sera carregado pela DataModel para um usuário, ambos passados como parametros para o Apache Mahout. Esta chamada permite que todos os processos da Univates que queiram exibir recomendações possam simplesmente fazer a chamada desta função, informando os parâmetros necessários. Caso a Univates queira

recomendar algum outro serviço, pode-se criar uma nova fonte de dados com os itens que se deseja recomendar e fazer uma nova chamada para esta função, passando como parâmetro a nova fonte de dados e a pessoa que receberá a recomendação.

Figura 21 – Função buscaRecomendacoes

```
public function buscaRecomendacoes( $cod_pessoa, $fonte_dado)
{
    TTransaction::open(self::$database);


    exec("java -jar /home/bugs/eclipse-workspace/recommender/target/
        recommender-0.0.1-SNAPSHOT-jar-with-dependencies.jar $cod_pessoa ", $saida, $retorno);

    TTransaction::close();
    return $retorno ;
}
```

Fonte: Do Autor (2019).

5.4 Implementação da Interface de Avaliação

Para a validação do protótipo, foi solicitado um backup das tabelas “processo_inscricao”, que contém todos os eventos cadastrados no sistema de inscrições da Univates, e a tabela “inscricao_processo”, onde estão as inscrições dos usuários nos eventos. Com isso foi criada uma base de dados que simula o sistema de inscrições da Univates.

O sistema de inscrições da Univates é onde são cadastrados os processos e realizada a inscrição de pessoas nesses processos, tais como inscrição de vestibular, aluguel da sede da univates, shows, palestras e semanas acadêmicas.

Na interface de gestão de fontes de dados, foi cadastrada uma conexão para a base de dados que contêm as inscrições; também foi feito o cadastro da fonte de dados com a consulta SQL para buscar as inscrições do sistema. A fonte de dados foi programada para ser reprocessada às 04:00, as 12:00 e as 20:00, de modo a atualizar o arquivo incluindo possíveis novas inscrições.

A fonte de dados é a responsável por criar o arquivo para a DataModel, trazendo o código do usuário inscrito e o código do evento, filtrando somente inscrições que não foram canceladas, pois a inscrição cancelada não deve ser considerada como preferência do usuário. Além disso a consulta busca somente processos que podem ser divulgados, evitando assim que seja recomendado algum processo que não possa ser divulgado. O arquivo gerado por

esta fonte de dados ficou com 181.849 registros, contendo 42.657 usuários distintos e 2.118 eventos distintos.

Ainda, para validação do protótipo, foi desenvolvida uma tela para exibição das recomendações para o usuário, permitindo que ele visualize e avalie recomendações, respondendo um breve questionário. Essa tela faz a chamada da Interface de gestão de fontes de dados via SOAP chamando a função “buscaRecomendacoes”, da classe “Recomendador”, passando os parâmetros o código da pessoa, e qual é a conexão que será utilizada, neste caso, a conexão de inscrições, como pode ser visto na Figura 22.

Figura 22 – Chamada via SOAP

```
try
{

    // define local do serviço
    $location = "http://165.227.9.145/recommender/soap.php?class=Recomendador";
    // instancia cliente SOAP
    $client = new SoapClient(NULL, array('encoding'   =>'UTF-8',
                                         'exceptions' => TRUE,
                                         'location'   => $location,
                                         'uri'        => "http://test-uri/"));
    // executa o método
    $recomendacoes = $client->buscaRecomendacoes('572090','todas_inscricoes'));

}
catch (Exception $e)
{
    echo 'Call error: ' . $e->getMessage();
}
```

Fonte: Do Autor (2019).

A interface com o questionário pode ser vista na Figura 23, nela o usuário digita o seu Código de aluno, este código é utilizado como referência do usuário dentro de todos os sistemas da univates, além de informar a idade, sexo e o curso que é formado ou está cursando. Após isso são apresentadas duas questões com o objetivo de validar o quanto o usuário percebe as recomendações feitas para ele, e o quanto ele acha válido um sistema de recomendações dentro da instituição Univates. Em seguida o usuário avalia três recomendações geradas para ele, atribuindo uma nota de 0 (irrelevante) até 10 (muito relevante), foi escolhido esta escala pois é a mesma escala que o Apache Mahout utiliza para exibir a força da recomendação gerada, permitindo assim uma comparação entre a nota do usuário e a força de recomendação retornada pelo recomendador.

Figura 23 – Questionário de avaliação

BEM-VINDO!

Este sistema gera recomendações de eventos baseado na similaridade do seu histórico de inscrições de eventos com os demais usuários da Universidade do Vale do Taquari Univates.

Este experimento possui viés acadêmico, visando a validação do sistema de recomendações desenvolvido para o trabalho de conclusão de curso do acadêmico de Engenharia da Computação Christian Rodolpho Bugs, portanto, suas respostas serão mantidas anônimas.

Código de Aluno

572090

Idade

Sexo ◦ Masculino
◦ Feminino

Curso

1) No seu dia-a-dia, você já observou o recebimento de recomendações de sistemas, sites ou aplicativos? Se sim, Quais?

2) Você considera que um sistema de recomendação para eventos de uma Universidade possa manter-lhe informado sobre potenciais interesses, de modo a contribuir com seu desenvolvimento acadêmico/profissional?

◦ Sim ◦ Não

Abaixo foram listadas recomendações de eventos para que você possa avaliar a relevância dos mesmos. É importante ressaltar, que na prática, o sistema recomendaria apenas eventos abertos, porém, a título de possibilitar uma validação rápida da ferramenta, foram incluindos eventos passados. Desta forma, pede-se que ao realizar a avaliação, você abstraia a temporalidade do evento e avalie-o conforme sua percepção de relevância para sua vida acadêmica/profissional.

3)

VI Semana Acadêmica do CGO

◦ 0 ◦ 1 ◦ 2 ◦ 3 ◦ 4 ◦ 5 ◦ 6 ◦ 7 ◦ 8 ◦ 9 ◦ 10

4)

9º CCTEC Congresso de Ciência e Tecnologia do Vale do Taquari

◦ 0 ◦ 1 ◦ 2 ◦ 3 ◦ 4 ◦ 5 ◦ 6 ◦ 7 ◦ 8 ◦ 9 ◦ 10

5)

SIPAT 2019

◦ 0 ◦ 1 ◦ 2 ◦ 3 ◦ 4 ◦ 5 ◦ 6 ◦ 7 ◦ 8 ◦ 9 ◦ 10

Neste campo você pode incluir sugestões, comentários ou críticas sobre o experimento e/ou sobre o sistema de recomendações.

Fonte: Do Autor (2019).

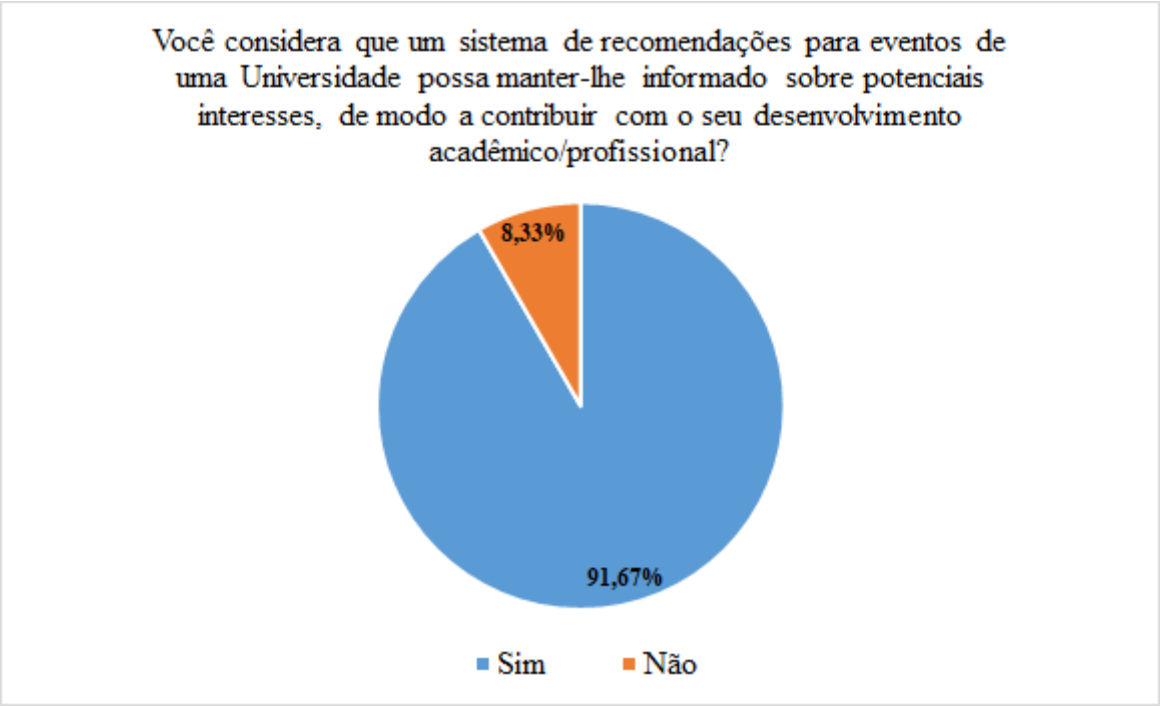
6 RESULTADOS E DISCUSSÃO

Foram realizados testes com usuários reais com o objetivo de validar o SR e avaliar a satisfação dos usuários com o conteúdo recomendado. Na realização da fase de testes participaram 12 pessoas, sendo 9 homens e 3 mulheres, cujo intervalo de idades foi desde os 19 até aos 32 anos (Média = 26; Desvio Padrão = 5,36). Os participantes da pesquisa foram estudantes e funcionários da Universidade Univates.

Na primeira questão do questionário foi perguntado: “No seu dia a dia você já observou o recebimento de recomendações de sistemas, sites ou aplicativos? Se sim, quais?”. Analisando os resultados é possível perceber que todos usuários já receberam recomendações vindas de um SR (100%). Dentre os SR que os usuários citaram conhecer estavam redes sociais e sites de vendas, porém nenhum dos entrevistados mencionou ter recebido recomendações de eventos. Este resultado pode estar relacionado ao fato de que a maioria dos usuários estuda ou trabalha na área de tecnologia da informação, tendo conhecimento sobre o que é um SR e também tendo uma grande frequência de utilização de conteúdos digitais.

Na sequência do questionário perguntou-se: “Você considera que um sistema de recomendações para eventos de uma Universidade possa manter-lhe informado sobre potenciais interesses, de modo a contribuir com o seu desenvolvimento acadêmico/profissional?. O objetivo desta pergunta foi o de saber o grau de interesse dos usuários em um SR específico para eventos universitários. O Gráfico 1 mostra os resultados obtidos, sendo que 91,67% dos entrevistados respondeu afirmativamente.

Gráfico 1 – Respostas da questão 2



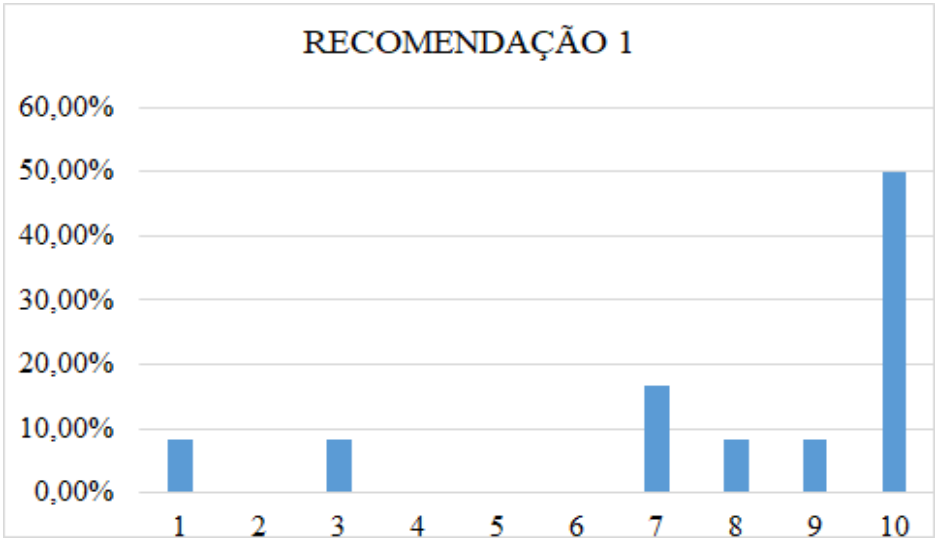
Fonte: Do Autor (2019).

Com os dados apresentados no Gráfico 1 é possível observar que a grande maioria considera que um SR para eventos de uma Universidade pode trazer informações que contribuam com o seu desenvolvimento acadêmico/profissional.

A próxima etapa dos testes ocorreu através da simulação de 3 recomendações de eventos para cada usuário, para cada recomendação sugerida era permitido atribuir uma pontuação que de 1 a 10. A título de possibilitar uma validação rápida da ferramenta, foram incluídos tanto eventos com inscrições abertas quanto eventos passados. A escolha por exibir eventos que já passaram, dá-se pelo fato de a base utilizada para os testes não ser atualizada com as novas inscrições que são feitas, assim os eventos que estão abertos para inscrição e não contém muitos inscritos podem ter um baixo valor de recomendação, pois este valor tende a crescer quanto mais pessoas estiverem inscritas no evento.

O Gráfico 2, apresenta os resultados das avaliações que os usuários fizeram para a Recomendação 1 que lhes foi sugerida.

Gráfico 2 – Resultados da recomendação 1



Fonte: Do Autor (2019).

Analisando o Gráfico 2 é possível observar que 50,02% dos usuários avaliaram com a pontuação máxima a Recomendação 1. Outro dado é que 16,66 % dos usuários avaliaram a recomendação com a pontuação de 7. E apenas 8,33% atribuíram a pontuação mínima a Recomendação 1. A pontuação média para a Recomendação 1 foi de 7,92 com desvio padrão de 3,02. Esse resultado demonstra que no geral as recomendações agradaram os usuários, e que a metade deles considerou a Recomendação 1 extremamente adequada.

O Gráfico 3 apresenta os resultados das avaliações que os usuários fizeram para a Recomendação 2 que lhes foi sugerida.

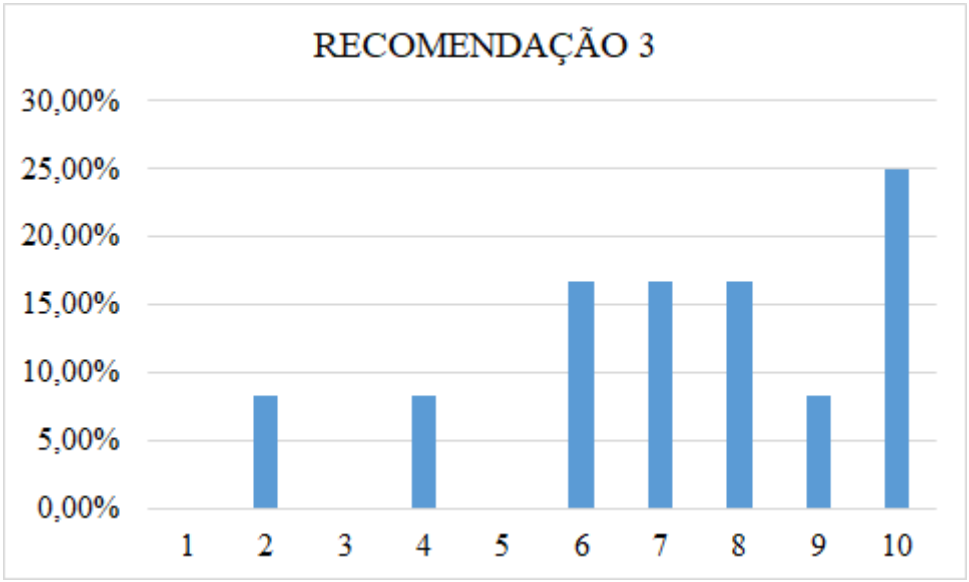
Gráfico 3 – Resultados da recomendação 2

Fonte: Do Autor (2019).

Em relação ao Gráfico 3 é possível observar que 50,02% dos usuários avaliaram com a pontuação máxima a Recomendação 2, dados similares ao da Recomendação 1. Também pode-se notar que 16,66 % dos usuários avaliaram a recomendação com a pontuação de 9. E nenhum usuário atribuiu a pontuação mínima. A pontuação média para a Recomendação 2 foi de 8,17 com desvio padrão de 2,69. Os resultados para a Recomendação 2 demonstraram-se altamente satisfatórios, levando em conta que 75% dos usuários avaliou com pontuação de 8 a 10 e que a metade avaliou com pontuação mais alta.

O Gráfico 4, abaixo, apresenta os resultados das avaliações que os usuários fizeram para a Recomendação 3 que lhes foi sugerida.

Gráfico 4 – Resultados da recomendação 3



Fonte: Do Autor (2019).

No Gráfico 4 observa-se uma maior distribuição entre as pontuações, sendo que 25,02% dos usuários avaliaram com a pontuação máxima a Recomendação 3. Também pode-se notar que 8,33% dos usuários avaliaram a recomendação com a pontuação de 9 e que para as pontuações 8, 7 e 6 houve a mesma porcentagem de avaliações (16,66%). A pontuação média para a Recomendação 3 foi de 7,25 com desvio padrão de 2,49. Os resultados para a Recomendação 3 foram inferiores aos encontrados para as Recomendações 1 e 2, no entanto ainda são considerados satisfatórios. Um ponto a ser destacado é que 50% dos usuários atribuíram notas entre 8 a 10 para a Recomendação 3, o que leva a crer que a consideraram adequada.

Como resultado final obteve-se 7,78 como pontuação média considerando os resultados para as três recomendações, sendo o desvio padrão de 0,19. Levando em consideração a escala de 1 a 10 pode-se dizer que o grau de avaliação das recomendações ficou em 77.80%, indicando que a grande maioria das pessoas que utilizaram o SR julgaram adequadas às recomendações propostas. Para melhor fundamentar a discussão dos resultados encontrados foi feita uma análise comparativa com alguns trabalhos semelhantes encontrados na literatura.

Na última etapa do questionário foi aberto um espaço para que os usuários deixassem comentários e observações sobre o sistema. Alguns usuários comentaram que os eventos nem sempre são amplamente divulgados e que o sistema ajudaria nesse ponto. Um usuário em específico sugeriu para que juntamente com o nome do evento recomendado aparecesse uma breve descrição do tema do evento. Outros usuários relataram gostar do sistema e das recomendações que receberam. Um dos usuários pediu para que o sistema fosse colocado em funcionamento pela Universidade Univates.

7 CONCLUSÕES

Durante o desenvolvimento deste trabalho foram estudados diversos tipos de filtragem de dados, algoritmos do Apache Mahout e abordagens relacionados à área de sistemas de recomendação. Através dessas pesquisas e do estudo de trabalhos relacionados foi possível concluir que a maneira mais eficiente de filtrar os dados referentes aos eventos da Universidade Univates a fim de gerar recomendações é utilizando filtragem colaborativa baseada no usuário. Como não existe na instituição um sistema que avalie a preferência dos alunos pelos eventos ofertados, a maneira mais adequada de realizar as recomendações é através da comparação de usuários similares. A grande variedade de eventos, cursos, oficinas, palestras e atividades ofertadas pela instituição também corrobora para a qualidade do banco de dados utilizado no presente trabalho.

Pode-se concluir também que a arquitetura disponibilizada pelo Apache Mahout é extremamente adequada ao tipo de SR proposto, disponibilizando uma documentação completa capaz de auxiliar tanto nas etapas de desenvolvimento quanto nas etapas de testes de performance do protótipo.

Ao final dos testes com usuários reais ficou evidente que os usuários compreendem a necessidade de um SR para os eventos da Univates, muitos relataram não participar das atividades em decorrência da falta de divulgação das mesmas. Dessa maneira constata-se que se este protótipo fosse de fato implementado aumentaria o acesso a divulgação para os alunos, consequentemente geraria um maior número de inscritos nos eventos ofertados.

Com relação às avaliações feitas para as 3 recomendações geradas o resultado de pontuação média de 7,78, em uma escala de 1 a 10, demonstra a aceitação dos usuários ao

sistema proposto. Além disso as Recomendações 1 e 2 foram avaliadas com nota máxima por metade dos usuários, indicando que em uma situação real os mesmos possivelmente se inscreveram nas atividades recomendadas.

7.1 Trabalhos Futuros

As avaliações das recomendações geradas pelo protótipo mostraram bons resultados, porém, ainda são necessárias algumas melhorias para que o sistema possa ser de fato implementado pela Univates. Poderia ser realizada uma análise comparativa com os eventos que não foram recomendados para os usuários alvos, validando a sua não-recomendação

Sugere-se como trabalhos futuros que o Apache Mahout seja iniciado como um serviço, pré-calculando as recomendações e mantendo em memória os resultados, de modo a reduzir o tempo de resposta às chamadas feitas pela interface que está consumindo o serviço.

Outra questão que pode ser abordada futuramente é a problemática de que o SR não é capaz de gerar recomendação para usuários que nunca se inscreveram em um evento, o que afetaria novos alunos da instituição, por exemplo. Nesta mesma linha, outra limitação do protótipo a ser revista é a impossibilidade de gerar recomendações para eventos que ainda não tenham nenhum inscrito. Para resolver essas limitações se fazem necessárias melhorias que visem incorporar ao SR a filtragem baseada em item.

REFERÊNCIAS

ADIANTI. **Adianti Framework**. Disponível em: <http://www.adianti.com.br/>. Acesso em: 25 de maio, 2019.

AGGARWAL C. C. **Recommender systems**: The Textbook**.** New York: Springer International Publishing, 2016. p. 1 - 28.

AGUIAR, J.; FECHINE, J.; COSTA, E. **Recomendação de Objetos de Aprendizagem utilizando Filtragem Colaborativa baseada em Tendências e em Estilos de Aprendizagem.** In: Simpósio Brasileiro de Informática na Educação-SBIE. 28, 2018. **Anais…** Rio de Janeiro, 2018. 1423 p.

ALVAREZ, E. B.; SIRIANI, A. L. R.; VIDOTTI, S. A. B. G.; CARVALHO, A. M. G. Os Sistemas de Recomendação, Arquitetura da Informação e a Encontrabilidade da Informação**. TransInformação**, v. 28, n. 3, p. 275-286, 2016.

AMARAL, F. **Introdução à Ciência de Dados:** mineração de dados e big data. Rio de Janeiro: Alta Books Editora, 2016.

APACHE. **Apache**. Disponível em: <https://www.apache.org/foundation/>. Acesso em: 26 de maio, 2019.

BARROS, P. A. **Sistema de filtragem colaborativa: um estudo de caso para venda e recomendação de marmitas**. 2014, 79 f. Monografia (Graduação) – Curso de Sistemas de Informação, Universidade do Planalto Catarinense, Lages, 2014.

BUSATTO, C. J. C**. O que está valendo?: um sistema web de recomendação de eventos**. 2013, 68 f. Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013.

CANHOTO, V. **Recomendação de música: comparação entre collaborative filtering e context filtering**. 2013, 95 f. Tese (Doutorado) – Programa de Pós-Graduação em Gestão de Sistemas de Informação, Instituto Universitário de Lisboa, Lisboa, 2013.

CARDOSO, M. B. **Movie. Me: um sistema de recomendação de filmes baseado no facebook**. 2016, 276 f. Monografia (Graduação) – Curso de Sistemas de Informação, Universidade Federal de Santa Catarina, Florianópolis, 2016.

CHENG, H.; KOC, L.; HARMSEN, J.; SHAKED, T.; CHANDRA, T.; ARADHYE, H.; ANDERSON, G.; CORRADO, G.; CHAI, W.; ISPIR, M.; ANIL, R.; HAQUE, Z.; HONG, L.; JAIN, V.; LIU, X.; SHAH, H. Wide & deep learning for recommender systems.

Proceedings of the 1st workshop on deep learning for recommender systems, p. 7-10, 2016.

COSTA, E.; AGUIAR, J.; MAGALHÃES, J. Sistemas de Recomendação de Recursos Educacionais: conceitos, técnicas e aplicações. In: MELO, A. M.; BORGES, M. A. F.; SILVA, C. G. (Org.). **Anais da II Jornada de Atualização em Informática na Educação.** Brasília: JAIE, 2019.

DA CONCEIÇÃO, F. L. A.; PÁDUA, F. L. C.; LACERDA, A.; MACHADO, A. C.; DALIP, D. H. Multimodal data fusion framework based on autoencoders for top-N recommender systems. **Applied Intelligence**, v. 49, n. 9, p. 3267-3282, 2016.

DA ROSA FURLAN, L. A.; ZAMBERLAN, A. O.; VIEIRA, S. A. G.; CANAL, A. P. Desenvolvimento de um sistema de recomendação para Bibliotecas Digitais. **Disciplinarum Scientia| Naturais e Tecnológicas**, v. 19, n. 1, p. 87-104, 2018.

FERRARI, D. G.; SILVA, L. N. C. **Introdução a mineração de dados**. Editora Saraiva, 2017.

GIL, A. C. C**omo elaborar projetos de pesquisa.** 4. ed. São Paulo: Atlas S/A, 2002.

GONÇALVES, F. **Sistema de Recomendação para Séries de TV por Assinatura**. 2016, 29 f. Monografia (Graduação) – Curso d Sistemas de Informação, Universidade Federal de Santa Maria, Santa Maria, 2016.

GUNAWARDANA, A.; SHANI, G. **Evaluating recommender systems.** Recommender systems handbook. Springer, Boston, MA, 2015. p. 265-308.

GUY, I. **Social recommender systems**. In: *Recommender systems handbook*. Springer, Boston, MA, 2015. p. 511-543.

IMDb. **Find movies, TV Shows, Celebrities and more…** Disponível em: < https://www.imdb.com/user/ur103715499/?ref_=nb_usr_prof_0 >. Acesso em: 17 nov. 2019.

IMDB. **IMDb Mobile Site.** Disponível em: <https://www.imdb.com/?ref_=nv_home>. Acesso em: 30 ago. 2019.

INGERSOLL, G. **Introdução ao Apache Mahout**. Disponível em: <https://www.ibm.com/developerworks/br/java/library/j-mahout/index.html#artrelatedtopics>. Acesso em: 28 de maio de 2019.

INTAGRAM. **Barra de busca**. Disponível em: < https://www.instagram.com/?hl=pt-br >. Acesso em: 17 nov. 2019.

LORENA, A. C.; CARVALHO, A. C. P. L. F. Uma introdução às support vector machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43-67, 2007.

MARCONI, M. A.; LAKATOS, E. M. **Fundamentos de Metodologia Científica.** 3. ed. São Paulo: Atlas, 2000.

MARQUES, G. F. A. **BOM APETITE - sistema de recomendação e apoio à decisão para dispositivos móveis**. 2016, 103 f. Tese (Doutorado) – Programa de Pós-Graduação em Engenharia Informática, Universidade de Lisboa, Lisboa, 2016.

MELVILLE, P., & SHINDHWANI V. **"**Recommender Systems". In: SAMMUT, C.; WEBB, G. (Eds.) **Encyclopedia of Machine Learning**. Berlin: Springer, 2010, pp. 829-838.

MENDES, T. M.; DOS SANTOS, R. V. M.; PICOLI, J. G. **Algoritmo de recomendação colaborativa aplicado ao contexto educacional/Algorithm of collaborative**

recommendation applied to the educational context. Brazilian Applied Science Review, 2018. p. 703-711.

NETSHOES. **Camisa Internacional I 19/20 s/nº Torcedor Nike Masculina - Vermelho e Branco**. Disponível em: <https://www.netshoes.com.br/camisa-internacional-i-1920-sn-torcedor-nike-masculina-vermelho+branco-HZM-1529-056>. Acesso em: 30 ago. 2019.

OWEN, S. **Mahout in action.** Shelter Island: Manning Publications Co. 2012.

PHP. 2019. Disponível em: <https://www.php.net/>. Acesso em 20 mar. 2019.

POSTGRESQL. **PostgreSQL:** The World's Most Advanced Open Source Relational Database. 2019. Disponível em: <https://www.postgresql.org/>. Acesso em: 27 de maio 2019.

PRODANOV, C. C; FREITAS, E. C. **Metodologia do trabalho científico:** métodos e técnicas da pesquisa e do trabalho acadêmico**.** 2. ed. Novo Hamburgo: Feevale, 2013.

QUILICI-GONZALEZ, J. A.; ZAMPIROLLI, F. de A. **Sistemas inteligentes e mineração de dados**. Santo André: Triunfa Gráfica e Editora, 2015.

RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha. **Recommender systems:** introduction and challenges. Boston: Springer, 2015.

RODRIGUES, F. R. **Sistemas de Recomendação para percursos culturais**. 2019, 56 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Informação, Universidade do Porto, Porto, 2019.

ROLIM, V.; FERREIRA, R.; COSTA, E.; CAVALCANTI, A.; DIONÍSIO, M. Um Estudo Sobre Sistemas de Recomendação de Recursos Educacionais. In: Congresso Brasileiro de Informática na Educação. 6, 2017. **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. Recife, 2017. 724 p.

SAMPAIO, I. A**. Apredizagem Ativa em Sistemas de Filtragem Colaborativa**. 2006, 107 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife, 2006.

SANTOS, A. A.; SANTOS, T. T. Detecção de frutos em campo por aprendizado de máquina. In: CONGRESSO INTERINSTITUCIONAL DE INICIAÇÃO CIENTÍFICA. Embrapa Informática Agropecuária, 2017. 11 p.

SANTOS, A. M. **Smart Marketing na TV Digital Interativa através de um sistema de recomendação de anúncios**. 2012, 143 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Gestão de Redes e Telecomunicação, Pontifícia Universidade Católica de Campinas, Campinas, 2012.

SCHAFER, J. B.; KONSTAN, J.; RIEDL, J. Recommender Systems. In: Proceedings of the 1st ACM conference on Electronic commerce, p. 158-166, 2000.

SOUZA FILHO, R. U. F. **Um sistema de recomendações de eventos culturais com áudio-descrição**. 2018, 68 f. Monografia (Graduação) – Curso de Sistemas de Informação, Universidade Federal Rural de Pernambuco, Recife, 2018.

TUDO GOSTOSO. **Macarrão com Queijo.** Disponível em: <https://www.tudogostoso.com.br/receita/83793-macarrao-com-requeijao.html>. Acesso em: 17 nov. 2019.